TURBOMOLE

Program Package for *ab initio*

Electronic Structure Calculations

USER'S MANUAL

TURBOMOLE Version 6.1
August 14, 2009

Contents

1 Preface and General Information **11**

1.1 Contributions and Acknowledgements 11

1.2 Features of TURBOMOLE 13

1.3 How to Quote Usage of TURBOMOLE 13

1.4 Modules and Their Functionality 21

1.5 Tools 23

2 Installation of TURBOMOLE **26**

3 How to Run TURBOMOLE **28**

3.1 A 'Quick and Dirty' Tutorial 28

3.1.1 Single Point Calculations: Running TURBOMOLE Modules . . 30

3.1.2 Energy and Gradient Calculations 30

3.1.3 Calculation of Molecular Properties 31

3.1.4 Modules and Data Flow 31

3.2 Parallel Runs 31

3.2.1 Running Parallel Jobs — MPI case 33

3.2.2 Running Parallel Jobs — OpenMP case 37

4 Preparing your input file with DEFINE **39**

4.0.1 Universally Available Display Commands in DEFINE 40

4.0.2 Specifying Atomic Sets 40

4.0.3 `control` as Input and Output File 40

4.0.4 Be Prepared 41

4.1 The Geometry Main Menu 42

4.1.1 Description of commands 44
4.1.2 Internal Coordinate Menu 47
4.1.3 Manipulating the Geometry 52
4.2 The Atomic Attributes Menu 52
4.2.1 Description of the commands 55
4.3 Generating MO Start Vectors 57
4.3.1 The MO Start Vectors Menu 57
4.3.2 Assignment of Occupation Numbers 60
4.3.3 Orbital Specification Menu 62
4.3.4 Roothaan Parameters 63
4.3.5 Start-MOs for broken symmetry treatments ("flip") 63
4.4 The General Options Menu 66
4.4.1 Important commands 67
4.4.2 Special adjustments 73
4.4.3 Relax Options 75
4.4.4 Definition of External Electrostatic Fields 79
4.4.5 Properties 80

5 Calculation of Molecular Structure and *Ab Initio* Molecular Dynamics 89

5.1 Structure Optimizations using the JOBEX Script 89
5.1.1 Options 89
5.1.2 Output 90
5.2 Program STATPT 91
5.2.1 General Information 91
5.2.2 Hessian matrix 92
5.2.3 Finding Minima 93
5.2.4 Finding transition states 93
5.3 Program Relax 94
5.3.1 Purpose 94
5.3.2 Optimization of General Coordinates 95
5.3.3 Force Constant Update Algorithms 96

5.3.4 Definition of Internal Coordinates 98
5.3.5 Structure Optimizations Using Internal Coordinates 98
5.3.6 Structure Optimization in Cartesian Coordinates 99
5.3.7 Optimization of Basis Sets (SCF only) 100
5.3.8 Simultaneous Optimization of Basis Set and Structure 100
5.3.9 Optimization of Structure and a Global Scaling Factor 101
5.3.10 Conversion from Internal to Cartesian Coordinates 101
5.3.11 Conversion of Cartesian Coordinates, Gradients and Force Constants to Internals 101
5.3.12 The m-Matrix 102
5.3.13 Initialization of Force Constant Matrices 102
5.3.14 Look at Results 103
5.4 Force Field Calculations 103
5.4.1 Purpose 103
5.4.2 How to Perform a UFF Calculation 104
5.4.3 The UFF implementation 104
5.5 Molecular Dynamics Calculations 106
5.6 Counterpoise-Corrections using the JOBBSSE Script 108
5.6.1 Options 109
5.6.2 Output 110

6 Hartree–Fock and DFT Calculations 111
6.1 Background Theory 113
6.2 Exchange-Correlation Functionals Available 114
6.3 Restricted Open-Shell Hartree–Fock 117
6.3.1 Brief Description 117
6.3.2 One Open Shell 117
6.3.3 More Than One Open Shell 120
6.3.4 Miscellaneous 122
6.4 Two-component Hartree–Fock and DFT Calculations 124
6.4.1 Background Theory 124
6.4.2 How to use 124

6.5 Using the Douglas–Kroll–Hess (DKH) Hamiltonian 126
6.6 Periodic Electrostatic Embedded Cluster Method 127
6.6.1 General Information 127
6.6.2 Theoretical Background 127
6.6.3 Calculation Setup 128
6.7 Empirical Dispersion Correction for DFT Calculations 135

7 Hartree–Fock and DFT Response Calculations: Stability, Dynamic Response Properties, and Excited States 138
7.1 Functionalities of Escf and Egrad 138
7.2 Theoretical Background 139
7.3 Implementation 141
7.4 How to Perform 142
7.4.1 Preliminaries 143
7.4.2 Polarizabilities and Optical Rotations 143
7.4.3 Stability Analysis 144
7.4.4 Vertical Excitation and CD Spectra 144
7.4.5 Excited State Geometry Optimizations 146
7.4.6 Excited State Force Constant Calculations 147
7.4.7 Polarizability Derivatives and Raman Spectra 147

8 Second-order Møller–Plesset Perturbation Theory 149
8.1 Functionalities of Mpgrad, Rimp2, Ricc2 149
8.2 Some Theory 150
8.3 How to Prepare and Perform MP2 Calculations 151
8.4 General Comments on MP2 Calculations, Practical Hints 153

9 Second-Order Approximate Coupled-Cluster (CC2) Calculations 155
9.1 CC2 Ground-State Energy Calculations 159
9.2 Calculation of Excitation Energies 161
9.3 First-Order Properties and Gradients 165
9.3.1 Ground State Properties, Gradients and Geometries 165
9.3.2 Excited State Properties, Gradients and Geometries 167

9.3.3 Visualization of densities and Density analysis 169
9.3.4 Fast geometry optimizations with RI-SCF based gradients . . 171
9.4 Transition Moments 172
9.5 RI-MP2-F12 Calculations 173
9.6 Parallel RI-MP2 and RI-CC2 Calculations 178
9.7 Spin-component scaling approaches (SCS/SOS) 179
9.7.1 Laplace-transformed SOS-RI-MP2 with $\mathcal{O}(\mathcal{N}^4)$ scaling costs . 180

10 Calculation of Vibrational Frequencies and Vibrational Spectra 183
10.1 Analysis of Normal Modes in Terms of Internal Coordinates 185
10.2 Calculation of Raman Spectra 186
10.3 Vibrational frequencies with fixed atoms using NumForce 186

11 Calculation of NMR Shieldings 188
11.1 Prerequisites 188
11.2 How to Perform a SCF of DFT Calculation 188
11.3 How to Perform a MP2 calculation 189
11.4 Chemical Shifts 190
11.5 Other Features and Known Limitations 190

12 Molecular Properties, Wavefunction Analysis, and Interfaces to Visualization Tools 191
12.1 Wavefunction analysis and Molecular Properties 191
12.2 Interfaces to Visualization Tools 193

13 Treatment of Solvation Effects with Cosmo 198

14 Keywords in the `control` file 203
14.1 Introduction 203
14.2 Format of Keywords and Comments 203
14.2.1 General Keywords 203
14.2.2 Keywords for System Specification 205
14.2.3 Keywords for redundant internal coordinates in `$redund_inp` 207
14.2.4 Keywords for Module Uff 209

14.2.5 Keywords for Modules Dscf and Ridft 213
14.2.6 Keywords for Periodic Electrostatic Embedded Cluster Method 234
14.2.7 Keywords for Cosmo 236
14.2.8 Keywords for Modules Grad and Rdgrad 239
14.2.9 Keywords for Module Aoforce 240
14.2.10 Keywords for Module Escf 243
14.2.11 Keywords for Module Egrad 245
14.2.12 Keywords for Modules Mpgrad and Rimp2 246
14.2.13 Keywords for Module Ricc2 249
14.2.14 Keywords for Module Relax 259
14.2.15 Keywords for Module Statpt 267
14.2.16 Keywords for Module Moloch 269
14.2.17 Keywords for wave function analysis and generation of plotting data 273
14.2.18 Keywords for Module Frog 280
14.2.19 Keywords for Module Mpshift 285
14.2.20 Keywords for Parallel Runs 287

15 Sample `control` files **291**
15.1 Introduction 291
15.2 NH_3 Input for a RHF Calculation 292
15.3 NO_2 input for an unrestricted DFT calculation 296
15.4 $TaCl_5$ Input for an RI-DFT Calculation with ECPs 300
15.5 Basisset optimization for Nitrogen 307
15.6 ROHF of Two Open Shells 310

16 The Perl-based Test Suite Structure **313**
16.1 General 313
16.2 Running the tests 313
16.3 Taking the timings and benchmarking 315
16.4 Modes and options of the TTEST script 315

Bibliography **319**

Index **328**

Chapter 1

Preface and General Information

1.1 Contributions and Acknowledgements

TURBOMOLE [1] is a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH 1989-2007, TURBOMOLE GmbH, since 2007. The following people have made contributions:

Reinhart Ahlrichs, Markus Klaus Armbruster, Rafał A. Bachorz, Michael Bär, Hans–Peter Baron, Rüdiger Bauernschmitt, Florian A. Bischoff, Stephan Böcker, Nathan Crawford, Peter Deglmann, Michael Ehrig, Karin Eichkorn, Simon Elliott, Filipp Furche, Andreas Glöß, Frank Haase, Marco Häser, Christof Hättig, Arnim Hellweg, Sebastian Höfener, Hans Horn, Christian Huber, Uwe Huniar, Marco Kattannek, Wim Klopper, Andreas Köhn, Christoph Kölmel, Markus Kollwitz, Klaus May, Paola Nava, Christian Ochsenfeld, Holger Öhm, Holger Patzelt, Dmitrij Rappoport, Oliver Rubner, Ansgar Schäfer, Uwe Schneider, Marek Sierka, David P. Tew, Oliver Treutler, Barbara Unterreiner, Malte von Arnim, Florian Weigend, Patrick Weis, Horst Weiss, Nina Winter

We acknowledge help from

- Michael Dolg, University of Stuttgart, now: University of Cologne
- Jürgen Gauss, University of Mainz
- Christoph van Wüllen, University of Bochum, now: TU Kaiserslautern
- Stefan Brode, BASF AG, Ludwigshafen
- Heinz Schiffer, HOECHST AG, Frankfurt

and financial support by the University of Karlsruhe, BASF AG, BAYER AG, HOECHST AG, the DFG, and the "Fonds der Chemischen Industrie".

Contact address:

Lehrstuhl für Theoretische Chemie
Institut für Physikalische Chemie
Universität Karlsruhe
Kaiserstr. 12
D-76128 Karlsruhe
E-mail: info@turbomole.com
Web: `http://www.turbomole.com`

Support is provided by COSMO*logic* GmbH&Co.KG, see `http://www.cosmologic.de`
Email: turbomole@cosmologic.de

1.2 Features of TURBOMOLE

TURBOMOLE has been specially designed for UNIX workstations and PCs and efficiently exploits the capabilities of this type of hardware. TURBOMOLE consists of a series of modules; their use is facilitated by various tools.

Outstanding features of TURBOMOLE are

- semi-direct algorithms with adjustable main memory and disk space requirements
- full use of all point groups
- efficient integral evaluation
- stable and accurate grids for numerical integration
- low memory and disk space requirements

1.3 How to Quote Usage of TURBOMOLE

Please quote the usage of the program package under consideration of the version number:

> TURBOMOLE V6.1 2009, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007,
> TURBOMOLE GmbH, since 2007; available from
> http://www.turbomole.com.

A LaTeX template could look like this:

```
@misc{TURBOMOLE,
title = {{TURBOMOLE V6.1 2009}, a development of {University of Karlsruhe} and
         {Forschungszentrum Karlsruhe GmbH}, 1989-2007,
         {TURBOMOLE GmbH}, since 2007; available from \\
         {\tt http://www.turbomole.com}.}}
```

Scientific publications require proper citation of methods and procedures employed. The output headers of TURBOMOLE modules include the relevant papers. One may also use the following connections between: method [module] number in the subsequent list (For module ricc2 see also Section 9).

- Programs and methods
  - general program structure and features: I

  - HF-SCF [`dscf`, `ridft`]: II
  - DFT (quadrature) [`dscf`, `ridft`, `escf`, `aoforce`]: IV, d (m grids)
  - RI-DFT [`ridft`, `aoforce`, `escf`]: c, d, XXIII (marij), VII (`escf`), XXIV (`aoforce`)
  - MP2 [`mpgrad`]: III
  - RI-MP2 [`rimp2`]: VIII, f
  - stability analysis [`escf`]: V
  - electronic excitations by CIS, RPA, TD-DFT [`escf`]: VI, VII, XVIII, XXVII
  - excited state structures and properties with CIS, RPA, TD-DFT [`egrad`]: XIX, XXVI, XXVII
  - RI-CC2 [`ricc2`]: XII,XIII (triplet excitations),XIV (properties for triplet states),XV (transition moments and properties of excited states),XXI (ground state geometry optimizations), XXII (excited state geometry optimizations and orbital-relaxed properties), XXVIII (parallelization)
  - analytical second derivatives (force fields) [`aoforce`]: XVI, XVII
  - RI-JK [`ridft`]: XX
  - NMR chemical shifts [`mpshift`]: IX (MP2)
  - parallel DFT [`ridft`]: X
  - geometry optimization in redundant internal coordinates [`relax`]: XI
  - RI integral evaluation: XXV

- Orbital and auxiliary basis sets
  - basis sets:
    * SV, SV(P), SVP, DZ (a), TZV, TZVP, TZVPP (b), TZVPP(Rb-Hg) (f), QZV, QZVP, QZVPP (i)
    * new balanced basis sets (with smaller ECPs, i.e. the def2 basis sets): j
    * all-electron basis sets for Rb to Xe (SVPall, SVPPall, TZVPall, TZVPPall): g
    * references for the correlation consistent basis sets (cc-pVXZ, etc.) can be found e.g. at `http://tyr0.chem.wsu.edu/~kipeters/Pages/cc_append.html` or `http://www.emsl.pnl.gov/forms/basisform.html`.
      Note, that most of the correlation consistent basis sets in the basis set library of `TURBOMOLE` have been downloaded from the latter EMSL web site and therefore users are requested to include in addition to the original scientific reference an appropriate citation (see web site) in any publications resulting from the use of these basis sets.
  - auxiliary basis sets for RI-DFT: c, d, e

– auxiliary basis sets for RI-MP2: f, k, h (for Dunning basis sets)

Further references of papers not from the TURBOMOLE group are given in the bibliography. The following publications describe details of the methodology implemented in TURBOMOLE:

Methods

I. Electronic Structure Calculations on Workstation Computers: The Program System TURBOMOLE. R. Ahlrichs, M. Bär, M. Häser, H. Horn and C. Kölmel; Chem. Phys. Letters **162**, 165 (1989).

II. Improvements on the Direct SCF Method. M. Häser and R. Ahlrichs; J. Comput. Chem. **10**, 104 (1989).

III. Semi-direct MP2 Gradient Evaluation on Workstation Computers: The MPGRAD Program. F. Haase and R. Ahlrichs; J. Comp. Chem. **14**, 907 (1993).

IV. Efficient Molecular Numerical Integration Schemes.
O. Treutler and R. Ahlrichs; J. Chem. Phys. **102**, 346 (1995).

V. Stability Analysis for Solutions of the Closed Shell Kohn–Sham Equation. R. Bauernschmitt and R. Ahlrichs; J. Chem. Phys. **104**, 9047 (1996).

VI. Treatment of Electronic Excitations within the Adiabatic Approximation of Time Dependent Density Functional Theory.
R. Bauernschmitt and R. Ahlrichs; Chem. Phys. Letters **256**, 454 (1996).

VII. Calculation of excitation energies within time-dependent density functional theory using auxiliary basis set expansions. R. Bauernschmitt, M. Häser, O. Treutler and R. Ahlrichs; Chem. Phys. Letters **264**, 573 (1997).

VIII. RI-MP2: first derivatives and global consistency. F. Weigend and M. Häser; Theor. Chem. Acc. **97**, 331 (1997).

IX. A direct implementation of the GIAO-MBPT(2) method for calculating NMR chemical shifts. Application to the naphthalenium and anthracenium ions. M. Kollwitz and J. Gauss; Chem. Phys. Letters **260**, 639 (1996).

X. Parallelization of Density Functional and RI-Coulomb Approximation in TURBOMOLE. M. v. Arnim and R. Ahlrichs; J. Comp. Chem. **19**, 1746 (1998).

XI. Geometry optimization in generalized natural internal Coordinates.
M. v. Arnim and R. Ahlrichs; J. Chem. Phys. **111**, 9183 (1999).

XII. CC2 excitation energy calculations on large molecules using the resolution of the identity approximation. C. Hättig and F. Weigend; J. Chem. Phys. **113**, 5154 (2000).

XIII. Implementation of RI-CC2 for triplet excitation energies with an application to *trans*-azobenzene. C. Hättig and Kasper Hald; Phys. Chem. Chem. Phys. **4** 2111 (2002).

XIV. First-order properties for triplet excited states in the approximated Coupled Cluster model CC2 using an explicitly spin coupled basis. C. Hättig, A. Köhn and Kasper Hald; J. Chem. Phys. **116**, 5401 (2002) and Vir. J. Nano. Sci. Tech., **5** (2002).

XV. Transition moments and excited-state first-order properties in the coupled-cluster model CC2 using the resolution-of-the-identity approximation. C. Hättig and A. Köhn; J. Chem. Phys. **117**, 6939 (2002).

XVI. An efficient implementation of second analytical derivatives for density functional methods. P. Deglmann, F. Furche and R. Ahlrichs; Chem. Phys. Letters **362**, 511 (2002).

XVII. Efficient characterization of stationary points on potential energy surfaces. P. Deglmann and F. Furche; J. Chem. Phys. **117**, 9535 (2002).

XVIII. An improved method for density functional calculations of the frequency-dependent optical rotation. S. Grimme, F. Furche and R. Ahlrichs; Chem. Phys. Letters **361**,321 (2002).

XIX. Adiabatic time-dependent density functional methods for excited state properties. F. Furche and R. Ahlrichs; J. Chem. Phys. **117**, 7433 (2002), J. Chem. Phys. **121**, 12772 (2004) (E).

XX. A fully direct RI-HF algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency. F. Weigend, Phys. Chem. Chem. Phys. **4**, 4285 (2002)

XXI. Geometry optimizations with the coupled-cluster model CC2 using the resolution-of-the-identity approximation. C. Hättig; J. Chem. Phys. **118**, 7751, (2003).

XXII. Analytic gradients for excited states in the coupled-cluster model CC2 employing the resolution-of-the-identity approximation. A. Köhn and C. Hättig; J. Chem. Phys., **119**, 5021, (2003).

XXIII. Fast evaluation of the Coulomb potential for electron densities using multipole accelerated resolution of identity approximation. M. Sierka, A. Hogekamp and R. Ahlrichs;J. Chem. Phys. **118**, 9136, (2003).

XXIV. Nuclear second analytical derivative calculations using auxiliary basis set expansion. P. Deglmann, K. May, F. Furche and R. Ahlrichs; Chem. Phys. Letters **384**, 103, (2004).

XXV. Efficient evaluation of three-center two-electron integrals over Gaussian functions. R. Ahlrichs; Phys. Chem. Chem. Phys. **6**, 5119, (2004).

XXVI. Analytical time-dependent density functional derivative methods within the RI-J approximation, an approach to excited states of large molecules. D. Rappoport and F. Furche, J. Chem. Phys. **122**, 064105 (2005).

XXVII. Density functional theory for excited states: equilibrium structure and electronic spectra. F. Furche and D. Rappoport, Ch. III of "Computational Photochemistry", Ed. by M. Olivucci, Vol. 16 of "Computational and Theoretical Chemistry", Elsevier, Amsterdam, 2005.

XXVIII. Distributed memory parallel implementation of energies and gradients for second-order Møller-Plesset perturbation theory with the resolution-of-the-identity approximation. Christof Hättig, Arnim Hellweg, Andreas Köhn, Phys. Chem. Chem. Phys. **8**, 1159-1169, (2006).

XXIX. Self-consistent treatment of spin-orbit interactions with efficient Hartree-Fock and density functional methods. Markus K. Armbruster, Florian Weigend, Christoph van Wüllen, Wim Klopper, Phys. Chem. Chem. Phys. **10**, 1748 - 1756, (2008).

Basis sets

The following tables can be used to find the proper citations of the standard orbital and auxiliary basis sets in the `TURBOMOLE` basis set library.

Orbital basis sets, elements H–Kr

	H,He	Li	Be	B–Ne	Na,Mg	Al–Ar	K	Ca	Sc–Zn	Ga–Kr
SVP,SV(P)	n	a	a	a	a	a	a	a	a	a
TZVP	n	b	b	b	b	b	b	b	b	b
TZVPP	n	f	f	f	f	f	f	f	f	f
QZVP,QZVPP	i									
def2-SV(P)	n	j	a	a	j	a	j	a	a	a
def2-SVP	n	j	a	a	j	a	j	a	j	a
def2-TZVP	n	f	j	f	j	j	j	f	j	f
def2-TZVPP	n	j	j	f	j	j	j	f	j	f

Note: For H–Kr def-SV(P), def-SVP, ... are identical with the basis sets without def prefix. def2-QZVPP and def2-QZVP are identical with QZVPP and QZVP.

Orbital basis sets, elements Rb–Rn

	Rb	Sr	Y–Cd	In–Cs	Ba	La–Hg	Tl–At	Rn
def-SVP,def-SV(P),def-TZVP	d	d	d	d	d	d	d	j
def-TZVPP	f	d	f	f	d	f	d	j
def2-SV(P)	j	d	d	j	d	d	j	j
def2-SVP	j	d	j	j	d	j	j	j
def2-TZVP,def2-TZVPP	j							
def2-QZVP,def2-QZVP	j							

Auxiliary basis sets for RI-DFT (Coulomb fitting)

	H–Kr	Rb–At	Rn
(def-)SVP,(def-)SV(P)	c	d	l
(def-)TZVP	d	d	l
def2 universal	l		

Auxiliary basis sets for RI-MP2 and RI-CC2, elements H–Kr

	H	He	Li	Be	B–F	Ne	Na,Mg	Al–Cl	Ar	K	Ca	Sc–Zn	Ga–Br	Kr
SVP,SV(P)	f	k	f	f	f	k	f	f	k	f	f	f	f	k
TZVP,TZVPP	f	k	f	f	f	k	f	f	k	f	f	f	f	k
QZVP,QZVPP	k													
def2-SV(P)	f	k	m	f	f	k	m	f	k	m	f	f	f	k
def2-SVP	f	k	m	f	f	k	m	f	k	m	f	m	f	k
def2-TZVP,def2-TZVPP	f	k	f	m	f	k	m	m	k	m	f	m	f	k
(aug-)cc-pVXZ, X=D–Q	h	h	k	k	h	h	k	h	h	-	-	-	h	h
(aug-)cc-pV5Z	k	k	-	-	k	k	-	k	k	-	-	-	-	-
cc-pWXZ, X=D–5	-	-	-	-	k	k	-	k	k	-	-	-	-	-

Note: the auxiliary basis sets for the (aug-)cc-pV(X+d)Z basis sets for Al–Ar are identical with the (aug-)cc-pVXZ auxiliary basis sets.

Auxiliary basis sets for RI-MP2 and RI-CC2, elements Rb–Rn

	Rb	Sr	Y–Cd	In–Cs	Ba	La–Hg	Tl–At	Rn
def-SVP,def-SV(P)	f							m
def2-SVP,def2-SV(P)	m	f	f	m	f	f	m	m
def-TZVP,def-TZVPP	f							m
def2-TZVP,def2-TZVPP	m							
def2-QZVP,def2-QZVP	m							

a. Fully Optimized Contracted Gaussian Basis Sets for Atoms Li to Kr.
A. Schäfer, H. Horn and R. Ahlrichs; J. Chem. Phys. **97**, 2571 (1992).

b. Fully Optimized Contracted Gaussian Basis Sets of Triple Zeta Valence Quality for Atoms Li to Kr. A. Schäfer, C. Huber and R. Ahlrichs; J. Chem. Phys. **100**, 5829 (1994).

c. Auxiliary Basis Sets to Approximate Coulomb Potentials.
K. Eichkorn, O. Treutler, H. Öhm, M. Häser and R. Ahlrichs; Chem. Phys. Letters **242**, 652 (1995).

d. Auxiliary basis sets for main row atoms and transition metals and their use to approximate Coulomb potentials. K. Eichkorn, F. Weigend, O. Treutler and R. Ahlrichs; Theor. Chem. Acc. **97**, 119 (1997).

e. Accurate Coulomb-fitting basis sets for H to Rn. F. Weigend; Phys. Chem. Chem. Phys. **8**, 1057 (2006).

f. RI-MP2: Optimized Auxiliary Basis Sets and Demonstration of Efficiency. F. Weigend, M. Häser, H. Patzelt and R. Ahlrichs; Chem. Phys. Letters **294**, 143 (1998).

g. Contracted all-electron Gaussian basis sets for Rb to Xe. R. Ahlrichs and K. May; Phys. Chem. Chem. Phys., **2**, 943 (2000).

h. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. F. Weigend, A. Köhn and C. Hättig; J. Chem. Phys. **116**, 3175 (2002).

i. Gaussian basis sets of quadruple zeta valence quality for atoms H–Kr.
F. Weigend, F. Furche and R. Ahlrichs; J. Chem. Phys. **119**, 12753 (2003).

j. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design an assessment of accuracy. F. Weigend and R. Ahlrichs; Phys. Chem. Chem. Phys. **7**, 3297 (2005).

k. Optimization of auxiliary basis sets for RI-MP2 and RI-CC2 calculation: Core-valence and quintuple-ζ basis sets for H to Ar and QZVPP basis sets for Li to Kr. C. Hättig; Phys. Chem. Chem. Phys. **7**, 59 (2005).

l. Accurate Coulomb-fitting basis sets for H to Rn. F. Weigend; Phys. Chem. Chem. Phys. **8**, 1057 (2006).

m. Optimized accurate auxiliary basis sets for RI-MP2 and RI-CC2 calculations for the atoms Rb to Rn. A. Hellweg, C. Hättig, S. Höfener and W. Klopper; Theor. Chem. Acc. **117**, 587 (2007).

n. unpublished.

1.4 Modules and Their Functionality

For references see Bibliography.

`define` interactive input generator which creates the input file `control`. `define` supports most basis sets in use, especially the only fully atom optimized consistent basis sets of SVP and TZV quality [2, 3, 4, 5, 6] available for the atoms H–Rn, excluding lanthanides. `define` determines the molecular symmetry and internal coordinates allowing efficient geometry optimization. `define` allows to perform a geometry optimization at a force field level to preoptimize the geometry and to calculate a Cartesian Hessian matrix. `define` sets the keywords necessary for single point calculations and geometry optimizations within a variety of methods. There are also many features to manipulate geometries of molecules: just try and see how it works.

`uff` performs a geometry optimization at a force field level. The Universal Force Field (UFF) [7] is implemented. Beyond this it calculates an analytical Hessian (Cartesian) which will be used as a start Hessian for an *ab initio* geometry optimization.

`dscf` for (semi–)direct SCF–HF and DFT calculations (see keywords for functionals supported). `dscf` supports restricted closed-shell (RHF), spin-restricted ROHF as well as UHF runs. `dscf` includes an in-core version for small molecules.

`grad` requires a successful `dscf` run and calculates the gradient of the energy with respect to nuclear coordinates for all cases treated by `dscf`.

`ridft` and `rdgrad` perform (direct) SCF–HF and DFT calculations—as `dscf` and `grad`—within the very efficient RI–J approximation for the interelectronic Coulomb term. These programs also permit to approximate HF exchange within the RI–K approximation. The exchange correlation functionals supported are specified in `define`.

`mpgrad` requires a well converged SCF run—by `dscf`, see keywords—and performs closed-shell RHF or UHF calculations yielding single point MP2 energies and, if desired, the corresponding gradient.

`rimp2` calculates MP2 energies and gradients for RHF and UHF wavefunctions, significantly more efficient than `mpgrad` by using the RI technique [8,9].

`ricc2` calculates electronic excitation energies, transition moments and properties of excited states at the CIS, CIS(D), ADC(2) and CC2 level using either a closed-shell RHF or a UHF SCF reference function. Calculates R12 basis set limit correction for MP2 energies. Employs the RI technique to approximate two-electron integrals. Includes as a subset also the functionalities of the `rimp2` program [10, 11, 12, 13].

`relax` requires a gradient run—by `grad`, `rdgrad`, `rimp2` or `mpgrad`—and proposes a new structure based on the gradient and the approximated force constants. The approximated force constants will be updated.

`statpt` performs structure optimization using the "Trust Radius Image Minimization" algorithm. It can be used to find minima or transition structures (first order saddle points). Transition structure searches usually require initial Hessian matrix calculated analytically or the transition vector from the lowest eigenvalue search.

`frog` executes one molecular dynamics (MD) step. Like `relax`, it follows a gradient run: these gradients are used as classical Newtonian forces to alter the velocities and coordinates of the nuclei.

`aoforce` requires a well converged SCF or DFT run—by `dscf` or `ridft`, see keywords—and performs an analytic calculation of force constants, vibrational frequencies and IR intensities. `aoforce` is also able to calculate only the lowest Hessian eigenvalues with the corresponding eigenvectors which reduces computational cost. The numerical calculation of force constants is also possible (see tool `Numforce` in Section 1.5).

`escf` requires a well converged SCF or DFT run and calculates time dependent and dielectric properties (spin-restricted closed-shell or spin-unrestricted open-shell reference):

- static and frequency-dependent polarizabilities within the SCF approximation
- static and frequency-dependent polarizabilities within the time-dependent Kohn–Sham formalism, including hybrid functionals such as B3-LYP
- electronic excitations within the RHF and UHF CI(S) restricted CI method
- electronic excitations within the so-called SCF-RPA approximation (poles of the frequency dependent polarizability)
- electronic excitations within the time dependent Kohn–Sham formalism (adiabatic approximation). It can be very efficient to use the RI approximation here, provided that the functional is of non-hybrid type: we recommend B-P86 (but slightly better results are obtained for the hybrid functional B3-LYP) [14].
- –stability analysis of single-determinant closed-shell wave functions (second derivative of energy with respect to orbital rotations) [15].

`egrad` computes gradients and first-order properties of excited states. Well converged orbitals are required. The following methods are available for spin-restricted closed shell or spin-unrestricted open-shell reference states:

– CI-Singles approximation (TDA)

– Time-dependent Hartree–Fock method (RPA)

– Time-dependent density functional methods

`egrad` can be employed in geometry optimization of excited states (using `jobex`, see Section 5.1), and in finite difference force constant calculations (using `Numforce`). Details see [16].

`mpshift` requires a converged SCF or DFT run for closed shells. `mpshift` computes NMR chemical shieldings for all atoms of the molecule at the SCF, DFT or MP2 level within the GIAO ansatz and the (CPHF) SCF approximation. From this one gets the NMR chemical shifts by comparison with the shieldings for the standard compound usually employed for this purpose, e.g. TMS for carbon shifts. Note that NMR shielding typically requires more flexible basis sets than necessary for geometries or energies. ECPs are not supported in `mpshift` [17].

`freeh` calculates thermodynamic functions from molecular data in a `control` file; an `aoforce` or a `NumForce` run is a necessary prerequisite.

1.5 Tools

Note: these tools are very helpful and meaningful for many features of `TURBOMOLE`.

This is a brief description of additional `TURBOMOLE` tools. Further information will be available by running the programs with the argument `-help`.

`Actual` please use: `actual -help`

`Aoforce2g98` usage: `aoforce2g98 aoforce.out > g98.out`
converts output from the `aoforce` program to Gaussian 98 style, which can be interpreted by some molecular viewer (e.g. jmol) to animate the normal coordinates.

`Bend` example: `bend 1 2 3`
displays the bending angle of three atoms specified by their number from the `control` file. Note that unlike in the `TURBOMOLE` definition of internal coordinates the apex atom is the second!

`Cbasopt` optimize auxiliary basis sets for RI-MP2 and RI-CC2 calculations. Uses `ricc2` to calculate the error functional and its gradient and `relax` as optimization module. For further details call `cbasopt -h`.

`Cgnce` plots energies as a function of SCF iteration number (gnuplot required).

`Cosmoprep` sets up `control` file for a `cosmo` run (see Chapter 13).

`Dist`	example: `dist 1 2` calculates atomic distances from `TURBOMOLE` input files; `dist -l 4` gives all interatomic distances to 4 a.u. (5 a.u. is the default).
`Eiger`	displays orbital eigenvalues obtained from data group `$scfmo`. Shows HOMO-LUMO gap, occupation, checks if there are holes in the occupation, and much more.
`Hcore`	prepares the `control` file for a Hamilton core guess (RHF only).
`Jobex`	usage: see Section 5.1 is the `TURBOMOLE` driver for all kinds of optimizations.
`Kdg`	example: `kdg scfdiis` kills a data group (here `$scfdiis`) in the `control` file.
`Lhfprep`	prepares lhf calculations by adjusting parameters of the `control` file.
`Log2x`	converts the file logging an MD trajectory into coordinates in frames appropriate for `jmol` animation program.
`Log2egy`	extracts the energy data (KE, total energy, PE) from an MD log file.
`Mdprep`	interactive program to prepare for an MD run, checking in particular the `mdmaster` file (`mdprep` is actually a FORTRAN program).
`Mp2prep`	prepares MP2 calculations interactively by adjusting parameters of the `control` file according to your system resources.
`Numforce`	calculates numerically force constants, vibrational frequencies, and IR intensities. (Note that the name of the shell script is `NumForce` with capital `F`.)
`Outp`	displays out-of-plan angles.
`Raman`	calculates vibrational frequencies and Raman intensities. See Section 10.2 for explanation.
`Screwer`	distorts a molecule along a vibrational mode.
`Sdg`	shows data group from `control` file: for example `sdg energy` shows the list of calculated energies.
`Sysname`	returns the name of your system, used in almost all `TURBOMOLE` scripts.
`Stati`	prepares the `control` file for a statistics run.
`t2x`	converts `TURBOMOLE` coordinates to xyz format.
`tm2molden`	creates a molden format input file for the Molden program. Molden is a graphical interface for displaying the molecular density, MOs, normal modes, and reaction paths. For more information about molden see: (`http://www.cmbi.ru.nl/molden/molden.html`).

`Tors` is a script to query a dihedral angle in a molecular structure: e.g. `tors 1 2 3 4` gives the torsional angle of atom 4 out of the plane of atoms 1, 2 and 3.

`Tbtim` is used to convert timings output files from TURBOBENCH calculations to LaTeXtables (for options please type `TBTIM --help`).

`Tblist` is used to produce summaries of timings from TURBOBENCH calculations to LaTeXformat. (for options please type `TBLIST --help`).

`Uhfuse` transforms the UHF MOs from a given symmetry to another symmetry, which is C_1 by default (just enter `uhfuse`). but can be specified (e.g. as C_{2v}) by entering `uhfuse -s c2v`. Now this functionality is included in the MO definition menu of `define` program, see Section 4.3.1.

`x2t` converts standard xyz files into `TURBOMOLE` coordinates.

Chapter 2

Installation of TURBOMOLE

Installation requires familiarity with some simple UNIX commands. The `TURBOMOLE` package is generally shipped as one *tar* file. This has to be uncompressed

```
gunzip turbomole.tar.gz
```

and unpacked

```
tar -xvf turbomole.tar
```

to produce the whole directory structure.

Note: Do not install or run `TURBOMOLE` as root or with root permissions!

Executable modules are in the `bin/[arch]` directory (for example, IBM modules are in `bin/rs6000-ibm-aix-5.2`). Tools (including `jobex`) are in `scripts` and (auxiliary) basis sets are kept in the directories `basen`, `jbasen`, `jkbasen` and `cbasen`. Coordinates for some common chemical fragments are supplied in `structures`. The documentation and a tutorial can be found in the folder `DOK`.

The environmental variable `$TURBODIR` must be set to the directory where `TURBOMOLE` has been unpacked, for example:

```
TURBODIR=/my_disk/my_name/TURBOMOLE
```

Check that the `Sysname` tool works on your computer:

```
$TURBODIR/scripts/sysname
```

should return the name of your system and this should match a `bin/`[arch] subdirectory.

If `Sysname` does not print out a single string matching a directory name in `$TURBODIR/bin/`, and if one of the existing binary versions does work, you can force `sysname` to print out whatever is set in the environment variable `$TURBOMOLE_SYSNAME`:

```
TURBOMOLE_SYSNAME=em64t-unknown-linux-gnu
```

Please make sure not to append `_mpi` to the string when setting `$TURBOMOLE_SYSNAME`, even if you intend to run parallel calculations. `sysname` will append this string automatically to the system name if `$PARA_ARCH` is set to `MPI` (see chapter 3.2.1 how to set up parallel environment).

You can call `TURBOMOLE` executables and tools easily from anywhere if you add the corresponding directories to your path (kornshell or bash):

```
PATH=$PATH:$TURBODIR/scripts
PATH=$PATH:$TURBODIR/bin/‘sysname‘
```

Now the `TURBOMOLE` executables can be called from a directory with the required input files. For example to call `dscf` and save the output:

```
$TURBODIR/bin/‘sysname‘/dscf > dscf.out
```

or if the path is OK, simply

```
dscf > dscf.out
```

In addition, some sample calculations are supplied in `Turbotest` so that the modules can be tested. Just run `TTEST` from this directory to run all tests or `TTEST -help` to get help on how this works.

Chapter 3

How to Run TURBOMOLE

3.1 A 'Quick and Dirty' Tutorial

All `TURBOMOLE` modules need the `control` file as input file. The `control` file provides directly or by cross references the information necessary for all kinds of runs and tasks (see Section 14). `define` provides step by step the `control` file: Coordinates, atomic attributes (e.g. basis sets), MO start vectors and keywords specific for the desired method of calculation. We recommend generating a set of Cartesian coordinates for the desired molecule using special molecular design software and converting this set into `TURBOMOLE` format (see Section 15.2) as input for `define`.

A straightforward way to perform a `TURBOMOLE` calculation from scratch is as follows:

- generate your atomic coordinates by any tool or program you are familiar with,
- save it as an `.xyz` file which is a standard output format of all programs, or use a conversion tool like `babel`,
- use the `TURBOMOLE` script `x2t` to convert your `.xyz` file to the `TURBOMOLE` `coord` file:
  ```
  x2t xyzinputfile > coord
  ```
- call `define`; after specifying the title, you get the coord menu—
  just enter `a coord` to read in the coordinates.
  Use `desy` to let `define` determine the point group automatically.
  If you want to do geometry optimizations, we recommend to use generalized internal coordinates; `ired` generates them automatically.
- you may then go through the menus without doing anything: just press `<Enter>`, `*` or `q`—whatever ends the menu, or by confirming the proposed decision of `define` again by just pressing `<Enter>`.
  This way you get the necessary specifications for a (SCF-based) run with SV(P) as the default basis set which is roughly 6-31G*.

- for more accurate SCF or DFT calculations choose larger basis sets, e.g. TZVP by entering `b all def-TZVP` or `b all def2-TZVP` in the basis set menu.
- ECPs which include (scalar) relativistic corrections are automatically used beyond Kr.
- an initial guess for MOs and occupation numbers is provided by `eht`
- for DFT you have to enter `dft` in the last menu and then enter `on`
- for efficient DFT calculations you best choose the RI approximation by entering `ri` and providing roughly 3/4 of the memory (with `m` *number*; *number* in MB) your computer has available. (Auxiliary basis sets are provided automatically) In the printout of an `ridft` run you can check how much is really needed; a `top` statement will tell you if you overplayed your cards.
- B-P86 is the default functional. It has a good and stable performance throughout the periodic system.
- for an HF or DFT run without RI, you simply enter:
  `[nohup] dscf > dscf.out &`
  or, for a RI-DFT run:
  `[nohup] ridft > ridft.out &`
- for a gradient run, you simply enter:
  `[nohup] grad > grad.out &`
  or
  `[nohup] rdgrad > rdgrad.out &`
- for a geometry optimization simply call `jobex`:
  for a standard SCF input:
  `[nohup] jobex &`
  for a standard RI-DFT input:
  `[nohup] jobex -ri &`
- many features, such as NMR chemical shifts on SCF and DFT level, do not require further modifications of the input, just call e.g. `mpshift` after the appropriate energy calculation (mpshift runs with SCF or DFT using a hybrid-functional need a file size of the semi-direct file `twoint` that is non-zero).
- other features, such as post–SCF methods need further action on the input, using either the last menu of define where one can activate all settings needed for DFT, TDDFT, MP2, CC2, etc. calculations (this is the recommended way), or tools like `Mp2prep` or `Rimp2prep`. Please refer to the following pages of this documentation.

3.1.1 Single Point Calculations: Running TURBOMOLE Modules

All calculations are carried out in a similar way. First you have to run `define` to obtain the `control` file or to add/change the keywords you need for your purpose. This can also be done manually with an editor. Given a bash and a path to `$TURBODIR/bin/`[arch] (see installation, Chapter 2) you call the appropriate module in the following way (e.g. module `dscf`):

```
nohup dscf > dscf.out &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. The output will be written to the file `dscf.out`. Several modules write some additional output to the `control` file. For the required keywords see Section 14. The features of `TURBOMOLE` will be described in the following section.

3.1.2 Energy and Gradient Calculations

Energy calculations may be carried out at different levels of theory.

Hartree–Fock–SCF
: use modules `dscf` and `grad` or `ridft` and `rdgrad` to obtain the energy and gradient. The energy can be calculated after a `define` run without any previous runs. `dscf` and `grad` need no further keywords `ridft` and `rdgrad` only need the keyword `$rij`. The gradient calculation however requires a converged `dscf` or `ridft` run.

Density functional theory
: DFT calculations are carried out in exactly the same way as Hartree–Fock calculations except for the additional keyword `$dft`. For DFT calculations with the fast Coulomb approximation you have to use the modules `ridft` and `rdgrad` instead of `dscf` and `grad`. Be careful: `dscf` and `grad` ignore RI–K flags and will try to do a normal calculation, but they will not ignore RI–J flags (`$rij`) and stop with an error message. To obtain correct derivatives of the DFT energy expression in `grad` or `rdgrad` the program also has to consider derivatives of the quadrature weights—this option can be enabled by adding the keyword `weight derivatives` to the data group `$dft`.

  For a semi-direct `dscf` calculation (Hartree–Fock or DFT) you first have to perform a statistics run. If you type

  ```
  stati dscf
  nohup dscf > dscf.stat &
  ```

  the disk space requirement (MB) of your current `$thime` and `$thize` combination will be computed and written to the data group `$scfintunit size=`*integer* (see Section 14.2.5). The requirement of other combinations will be computed as well and be written to the output file `dscf.stat`. The size of the integral

file can be set by the user to an arbitrary (but reasonable) number. The file will be written until it reaches the given size and `dscf` will continue in direct mode for the remaining integrals. Note that `TURBOMOLE` has no 2GB file size limit.

MP2
: MP2 calculations need well converged SCF runs (the SCF run has to be done with at least the density convergence `$denconv 1.d-7`, and `$scfconv 7` as described in Section 14). This applies to CC2 or SCS-MP2 also. For MP2 calculations in the RI approximation use the `ricc2` module. The input can be prepared with the `cc2` menu in `define`. (Alternatively, the older `rimp2` module and for preparation of its input the tool `Rimp2prep` maybe used). The module `mpgrad` calculates the canonical (non-RI) MP2 energy as well as the energy gradient. If only the energy is desired use the keyword `$mp2energy`. For all further preparations run the tool `Mp2prep`.

Excited states (`escf`)
: Single point excited state energies for CIS, TDHF, and TDDFT methods can be calculated using `escf`. Excited state energies, gradients, and other first order properties are provided by `egrad`. Both modules require well converged ground state orbitals.

Excited states (`ricc2`)
: The module `ricc2` calculates MP2 and CC2 ground state energies and CIS/CCS, CIS(D), CIS(D_∞), ADC(2) or CC2 excitation energies using the resolution-of-the-identity (RI) approximation. Excited state gradients are available at the CCS, CIS(D_∞), ADC(2), and CC2 levels. In addition, transition moments and first-order properties are available for some of the methods. For more details see Section 9. The input can be prepared using the `cc2` menu of `define`.

3.1.3 Calculation of Molecular Properties

See Section 1.4 for the functionality and Section 14 for the required keywords of the modules `dscf`, `ridft`, `mpshift`, `escf`, and `ricc2`.

3.1.4 Modules and Data Flow

See Figure 3.1.

3.2 Parallel Runs

Some of the `TURBOMOLE` modules are parallelized using the message passing interface (MPI) for distributed and shared memory machines or, in the case of `dscf` and `ricc2` also with OpenMP for shared memory machines. The list of parallelized programs includes presently:

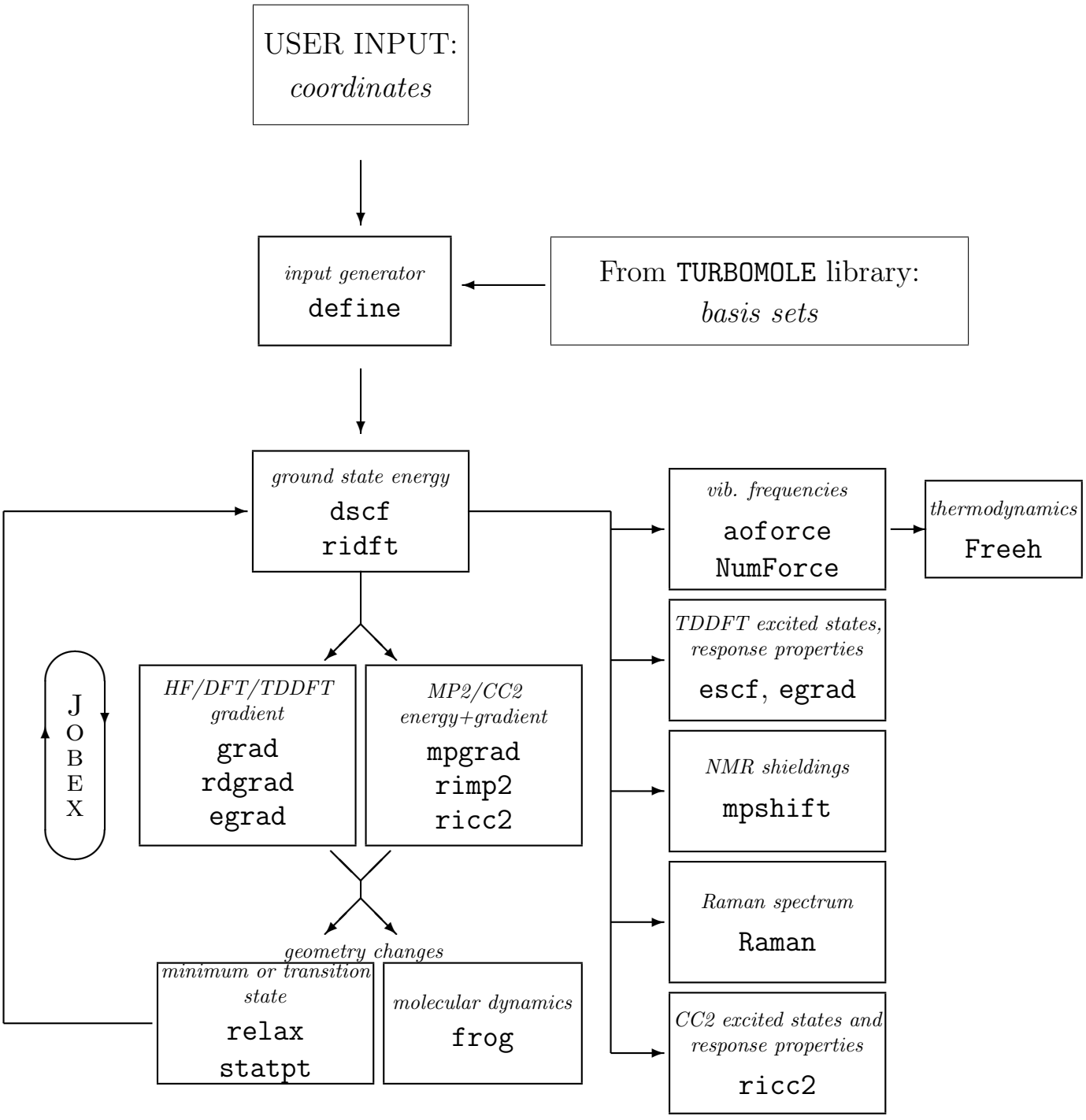

Figure 3.1: The modules of TURBOMOLE and the main data flow between them.

- `ridft` — parallel ground state non-hybrid DFT energies including RI-J and the multipole accelerated RI (MA-RI-J)
- `rdgrad` — parallel ground state gradients from `ridft` calculations
- `dscf` — Hartree-Fock and DFT ground state calculations for all available DFT functionals, without the usage of RI-J approximation
- `grad` — parallel ground state gradients from `dscf` calculations
- `ricc2` — parallel ground and excited state calculations of energies and gradients at MP2 and CC2 level using RI, as well as energy calculations of other wave function models, see chapter 9.6.
- `mpgrad` — parallel conventional (i.e. non-RI) MP2 energy and gradient calculations. Please note that RI-MP2 is one to two orders of magnitue faster than conventional MP2, so even serial RI-MP2 will be faster than parallel MP2 calculations.
- `NumForce` — this script can used for a trivial parallelization of the numerical displaced coordinates.

Additional keywords neccessary for parallel runs with the MPI binaries are described in Chapter 14. However, those keywords do not have to be set by the users. When using the parallel version of `TURBOMOLE`, scripts are replacing the binaries. Those scripts prepare a usual input, run the necessary steps and automatically start the parallel programs. The users just have to set environment variables, see Sec. 3.2.1 below.

To use the OpenMP parallelization only an environment variable needs to be set. But to use this parallelization efficiently one should consider a few additional points, e.g. memory usage, which are described in Sec. 3.2.2.

3.2.1 Running Parallel Jobs — MPI case

The parallel version of `TURBOMOLE` runs on all supported systems:

- workstation cluster with Ethernet, Infiniband, Myrinet (or other) connection
- SMP systems
- or combinations of SMP and cluster

Setting up the parallel environment

In addition to the installation steps described in Section 2 (see page 26) you just have to set the variable `PARA_ARCH` to `MPI`, i.e. in `sh/bash/ksh` syntax:

```
export PARA_ARCH=MPI
```

This will cause `sysname` to append the string `_mpi` to the system name and the scripts like `jobex` will take the parallel binaries by default. To call the parallel versions of the programs `ridft`, `rdgrad`, `dscf`, `grad`, `ricc2`, or `mpgrad` from your command line without explicit path, expand your `$PATH` environment variable to:

```
export PATH=$TURBODIR/bin/‘sysname‘:$PATH
```

The usual binaries are replaced now by scripts that prepare the input for a parallel run and start `mpirun` (or `poe` on IBM) automatically. The number of CPUs that shall be used can be chosen by setting the environment variable `PARNODES`:

```
export PARNODES=8
```

The default for `PARNODES` is 2.

Finally the user can set a default scratch directory that must be available on all nodes. Writing scratch files to local directories is highly recommended, otherwise the scratch files will be written over the network to the same directory where the input is located. The path to the local disk can be set with

```
export TURBOTMPDIR=/scratch/username/
```

This setting is automatically recognized by the parallel `ridft` and `ricc2` programs. Note that this does not set the path for the integer scratch files for `dscf` (see section below about `twoint` of keyword `$scfintunit`).

If `TURBOTMPDIR` is not set by the user, `ridft` will check for a `/tmp`, `/scr`, or `/work` directory. If one of those is found, it will be used by default.

On all systems `TURBOMOLE` is using the MPI library that has been shipped with your operating system.

On Linux for PCs and Itanium2 systems HP-MPI is used — see
`http://www.hp.com/go/mpi/`
The binaries that initialize MPI and start the parallel binaries (`mpirun`) are located in the `$TURBODIR/mpirun_scripts/HPMPI/` directory.

Note: the parallel `TURBOMOLE` modules (except `ricc2`) need an extra server running in addition to the clients. This server is included in the parallel binaries and it will be started automatically—but this results in one additional task that usually does not need any CPU time. So if you are setting `PARNODES` to `N`, `N+1` tasks will be started.

If you are using a queuing system or if you give a list of hosts where `TURBOMOLE` jobs shall run on (see below), make sure that the number of supplied nodes match `$PARNODES` — e.g. if you are using 4 CPUs via a queuing system, make sure that `$PARNODES` is set to 4.

On IBM systems the total number of tasks in the LoadLeveller script must be set to `$PARNODES + 1` except for `ricc2`.

Starting parallel jobs

After setting up the parallel environment as described in the previous section, parallel jobs can be started just like the serial ones. If the input is a serial one, it will be prepared automatically for the parallel run.

The parallel versions of the programs `dscf` and `grad` need an integral statistics file as input which is generated by a parallel statistics run. This preparation step is done automatically by the scripts `dscf` and `grad` that are called in the parallel version. In this preparing step the size of the file that holds the 2e-integrals for semi-direct calculations `twoint` is recalculated and reset. It is highly recommended to set the path of this `twoint` file to a local scratch directory of each node by changing the line.

```
unit=30 size=?????  file=twoint
```

to

```
unit=30 size=?????  file=/local_scratchdir/twoint
```

For the additional mandatory or optional input for parallel runs with the `ricc2` program see Section 9.6.

Running calculations on different nodes

If `TURBOMOLE` is supposed to run on a cluster, we highly recommed the usage of a queuing system like PBS. The parallel version of `TURBOMOLE` will automatically recognise that it is started from within the PBS environment and the binaries will run on the machines PBS provides for each job.

Important: Make sure that the input files are located on a network directory like an NFS disk which can be accessed on all nodes that participate at the calculation.

A file that contains a list of machines has to be created, each line containing one machine name:

```
node1
node1
node2
node3
node4
node4
```

And the environment variable `$HOSTS_FILE` has to be set to that file:

```
export HOSTS_FILE=/nfshome/username/hostsfile
```

Note: Do not forget to set `$PARNODES` to the number of lines in `$HOSTS_FILE`.

Note: In general the stack size limit has to be raised to a reasonable amount of the memory (or to ulimited). In the serial version the user can set this by `ulimit -s unlimited` on bash/sh/ksh shells or `limit stacksize unlimited` on csh/tcsh shells. However, for the parallel version that is not sufficient if several nodes are used, and the `/etc/security/limits.conf` files on all nodes might have to be changed. Please see the following web site for details: `Turbomole Forum`

Testing the parallel binaries

The binaries `ridft`, `rdgrad`, `dscf`, `grad`, and `ricc2` can be tested by the usual test suite: go to `$TURBODIR/TURBOTEST` and call `TTEST`

Note: Some of the tests are very small and will only pass properly if 2 CPUs are used at maximum. Therefore `TTEST` will not run any test if `$PARNODES` is set to a higher value than 2.

If you want to run some of the larger tests with more CPUs, you have to edit the `DEFCRIT` file in `TURBOMOLE/TURBOTEST` and change the `$defmaxnodes` option.

Linear Algebra Settings

The number of CPUs and the algorithm of the linear algebra part of Turbomole depends on the settings of `$parallel_platform`:

cluster – for clusters with TCP/IP interconnect. Communication is avoided by using an algorithm that includes only one or few CPUs.

MPP – for clusters with fast interconnect like Infiniband or Myrinet. Number of CPUs that take part at the calculation of the linear algebra routines depends on the size of the input and the number of nodes that are used.

SMP – all CPUs are used and SCALapack (see http://www.netlib.org/scalapack/) routines are involved.

The scripts in `$TURBODIR/mpirun_scripts` automatically set this keyword depending on the output of `sysname`. All options can be used on all systems, but especially the SMP setting can slow down the calculation if used on a cluster with high latency or small bandwidth.

Sample simple PBS start script

```
#!/bin/sh
# Name of your run :
#PBS -N turbomole
#
```

```
# Number of nodes to run on:
#PBS -l nodes=4
#
# Export environment:
#PBS -V

# Set your TURBOMOLE pathes:

######## ENTER YOUR TURBOMOLE INSTALLATION PATH HERE ##########
export TURBODIR=/whereis/TURBOMOLE
################################################################

export PATH=$TURBODIR/scripts:$PATH

## set locale to C
unset LANG
unset LC_CTYPE

# set stack size limit to unlimited:
ulimit -s unlimited

# Count the number of nodes
PBS_L_NODENUMBER=‘wc -l < $PBS_NODEFILE‘

# Check if this is a parallel job
if [ $PBS_L_NODENUMBER -gt 1 ]; then
##### Parallel job
# Set environment variables for a MPI job
    export PARA_ARCH=MPI
    export PATH="${TURBODIR}/bin/‘sysname‘:${PATH}"
    export PARNODES=‘expr $PBS_L_NODENUMBER‘
else
##### Sequentiel job
# set the PATH for Turbomole calculations
    export PATH="${TURBODIR}/bin/‘sysname‘:${PATH}"
fi

######## ENTER YOUR JOB HERE ###################################
jobex -ri > jobex.out
################################################################
```

3.2.2 Running Parallel Jobs — OpenMP case

The programs `dscf` and `ricc2` are also partially parallelized with OpenMP for applications on shared-memory, in particular multi-CPU and multi-core, machines.

The OpenMP parallelization does not need any special program startup. The binaries can be invoked in exactly the same manner as for sequential (non-parallel) calculations. The only difference is, that before the program is started the environment variable `OMP_NUM_THREADS` has to be set to the number of threads that should be used by the program. The number of threads is is essentially the max. number of CPU cores the program will try to utilize. To exploit e.g. all eight cores of a machine with two quad-core CPUs set

```
export OMP_NUM_THREADS=8
```

(for csh and tcsh use `setenv OMP_NUM_THREADS=8`).

Presently the OpenMP parallelization of `ricc2` comprises all functionalities apart from the recently implemented RI-MP2-F12, the LT-SOS-RI-MP2, and the calculation of expectation values for $\hat{S}^2$. Note that the memory specified with `$maxcor` is for OpenMP-parallel calculation the maximum amount of memory that will be dynamically allocated by all threads together. To use your computational resources efficiently, it is recommended to set this value to about 75% of the physical memory available for your calculations, but to at most 16000 (megabytes). (Due to the use of integer$*$4 arithmetics the `ricc2` program is presently limited to 16 Gbytes.)

In the `dscf` program the OpenMP parallelization covers presently only the Hartree-Fock coulomb and exchange contributions to the Fock matrix in fully integral-direct mode and is mainly intended to be used in combination with OpenMP parallel runs of `ricc2`. Nevertheless, the OpenMP parallelization can also be used in DFT calculations, but the numerical integration for the DFT contribution to the Fock matrix will only use a single thread (CPU core) and thus the overall speed up will be less good.

Restrictions:

- In the `ricc2` program the parts related to RI-MP2-F12, LT-SOS-RI-MP2 or calculation of expectation values for $\hat{S}^2$ to not (yet) use OpenMP parallelization. If the OpenMP parallelization is switched on (by setting `OMP_NUM_THREADS`) these parts will still be executed sequentially.
- In the `dscf` program the DFT part will only be executed sequentially by a single thread and the `$incore` option will be ignored if more than one thread is used. Semi-direct `dscf` calculations (i.e. if a size larger than 0 is given two-electron integral scratch file in `$scfintunit`) can not be combined with the OpenMP parallel runs. (The program will than stop with error message in the first Fock matrix construction.)

Chapter 4

Preparing your input file with DEFINE

`define` is the general interactive input generator of `TURBOMOLE`. During a session with `define`, you will create the `control` file which controls the actions of all other `TURBOMOLE` programs. During your `define` session you will be guided through four main menus:

1. **The geometry main menu**: This first menu allows you to build your molecule, define internal coordinates for geometry optimizations, determine the point group symmetry of the molecule, adjust internal coordinates to the desired values and related operations. Beyond this one can perform a geometry optimization at a force field level to preoptimize the geometry and calculate a Cartesian analytical Hessian. After leaving this menu, your molecule to be calculated should be fully specified.

2. **The atomic attributes menu**: Here you will have to assign basis sets and/or effective core potentials to all atoms. The SV(P) basis is assigned automatically as default, as well as ECPs (small core) beyond Kr.

3. **The occupation numbers and start vectors menu**: In this menu you should choose `eht` to start from Extended Hückel MO vectors. Then you have to define the number of occupied orbitals in each irreducible representation.

4. **The general menu**: The last menu manages a lot of control parameters for all `TURBOMOLE` programs.

Most of the menu commands are self-explanatory and will only be discussed briefly. Typing `*` (or `q`) terminates the current menu, writes data to `control` and leads to the next while typing `&` goes back to the previous menu.

4.0.1 Universally Available Display Commands in DEFINE

There are some commands which may be used at (almost) every stage of your `define` session. If you build up a complicated molecular geometry, you will find the `dis` command useful. It will bring you to the following little submenu:

```
ANY COMMAND WHICH STARTS WITH THE 3 LETTERS  dis  IS A
DISPLAY COMMAND. AVAILABLE DISPLAY COMMANDS ARE :
disc  <range> : DISPLAY CARTESIAN COORDINATES
dist  <real>  : DISPLAY DISTANCE LIST
disb  <range> : DISPLAY BONDING INFORMATION
disa  <range> : DISPLAY BOND ANGLE INFORMATION
disi  <range> : DISPLAY VALUES OF INTERNAL COORDINATES
disg  <range> : GRAPHICAL DISPLAY OF MOL. GEOMETRY
<range> IS A SET OF ATOMS REFERENCED
<real>  IS AN OPTIONAL DISTANCE THRESHOLD (DEFAULT=5.0)
AS AN EXAMPLE CONSIDER  disc  1,3-6,10,11  WHICH DISPLAYS
THE CARTESIAN COORDINATES OF ATOMS 1,3,4,5,6,10,and 11 .
HIT >return< TO CONTINUE OR ENTER ANY DISPLAY COMMAND
```

Of course, you may enter each of these display commands directly without entering the general command `dis` before. The option `disg` needs special adaption to the computational environment, however, and will normally not be available.

4.0.2 Specifying Atomic Sets

For many commands in `define` you will have to specify a set of atoms on which that command shall act. There are three ways to do that:

- You may enter `all` or `none`, the meaning of which should be clear (entering `none` makes not much sense in most cases, however).
- You may specify a list of atomic indices like `1` or `3,5,6` or `2,4-6,7,8-10` or similar.
- You may also enter atomic identifiers which means strings of at most eight characters: the first two contain the element symbol and the remaining six could be used to distinguish different atoms of the same type. For example, if you have several carbon atoms in your molecule, you could label some `c ring` and others `c chain` to distinguish them. Whenever you want to enter an *atomic identifier*, you have to put it in double quotation marks: `"c ring"`.

You should take into account that `define` also creates, from the atoms you entered, all others according to symmetry. If necessary, you will therefore have to lower the (formal) symmetry before executing a command.

4.0.3 `control` as Input and Output File

`define` may be used to update an existing `control` file, which is helpful if only the basis set has been changed. In this case just keep all data, i.e. reply with `<enter>` on

all questions, and only specify new start MOs. The more general usage is described now.

At the beginning of each `define` session, you will be asked to enter the name of the file to be created. As mentioned earlier, all `TURBOMOLE` programs require their input to be on a file named `control`, but it may be useful at this moment to choose another name for this file (e.g. if you have an old input file `control` and you do not want to overwrite it). Next you will be asked to enter the name of an *old* file which you want to use as input for this session. This prevents you from creating the new input from scratch if you want to make only minor changes to an old `control` file. It is possible to use the same file as input and output file during a `define` session (which means that it will only be modified). This may lead to difficulties, however, because `define` reads from the input file when entering each main menu and writes the corresponding data when leaving this menu. Therefore the input file may be in an ill-defined status for the next main menu (this will be the case, for example, if you add or change atoms in the first menu so that the basis set information is wrong in the second menu). `define` takes care of most—but not all—of these problems.

For these reasons, it is recommended to use a different filename for the input and the output file of the `define` session if you change the molecule to be investigated. In most cases involving only changes in the last three of the four main menus no problem should arise when using the same file as input and output.

4.0.4 Be Prepared

Atomic Coordinates

Molecules and their structures are specified by coordinates of its atoms, within the program invariably by Cartesian coordinates in atomic units (Ångstrøm would also do). In `TURBOMOLE` these coordinates are contained in the file `coord` (see Section 15 “Sample `control` files” for an example).

Recommendation

We strongly recommend to create the `coord` file *before* calling `define`, only for small molecules one should use the interactive input feature of `define`. Set up the molecule by any program you like and write out coordinates in the xyz-format (`XMol` format), which is supported by most programs. Then use the `TURBOMOLE` tool `x2t` to convert it into a `TURBOMOLE` `coord` file (see Section 1.5.

Internal Coordinates

Structure optimizations, see `jobex`, are most efficient if carried out in internal coordinates and `TURBOMOLE` offers the following choices.

`internals` based on bond distances and angles, see Section 4.1.2.

`redundant internals`
: defined as linearly independent combinations of *internals* (see ref. [18]), provided automatically by the command `ired` in the 'geometry main menu' in Section 4.1 below. This works in almost all cases and is efficient. The disadvantage is, that this is a black box procedure, the coordinates employed have no direct meaning and cannot be modified easily by the user.

`cartesians`
: should always work but are inefficient (more cycles needed for convergence). Cartesians are the last resort if other options fail, they are assigned as default if one leaves the main geometry menu and no other internals have been defined.

4.1 The Geometry Main Menu

After some preliminaries providing the title etc. you reach the geometry main menu:

```
SPECIFICATION OF MOLECULAR GEOMETRY ( #ATOMS=0     SYMMETRY=c1  )
YOU MAY USE ONE OF THE FOLLOWING COMMANDS :
sy <group> <eps> : DEFINE MOLECULAR SYMMETRY (default for eps=3d-1)
desy <eps>       : DETERMINE MOLECULAR SYMMETRY AND ADJUST
                     COORDINATES (default for eps=1d-6)
susy             : ADJUST COORDINATES FOR SUBGROUPS
ai               : ADD ATOMIC COORDINATES INTERACTIVELY
a <file>         : ADD ATOMIC COORDINATES FROM FILE <file>
aa <file>        : ADD ATOMIC COORDINATES IN ANGSTROEM UNITS FROM FILE <file>
sub              : SUBSTITUTE AN ATOM BY A GROUP OF ATOMS
i                : INTERNAL COORDINATE MENU
ired             : REDUNDANT INTERNAL COORDINATES
red_info         : DISPLAY REDUNDANT INTERNAL COORDINATES
ff               : UFF-FORCEFIELD CALCULATION
m                : MANIPULATE GEOMETRY
frag             : DEFINE FRAGMENTS FOR BSSE CALCULATION
w <file>         : WRITE MOLECULAR COORDINATES TO FILE <file>
r <file>         : RELOAD ATOMIC AND INTERNAL COORDINATES FROM FILE <file>
name             : CHANGE ATOMIC IDENTIFIERS
del              : DELETE ATOMS
dis              : DISPLAY MOLECULAR GEOMETRY
banal            : CARRY OUT BOND ANALYSIS
*                : TERMINATE MOLECULAR GEOMETRY SPECIFICATION
                     AND WRITE GEOMETRY DATA TO CONTROL FILE

IF YOU APPEND A QUESTION MARK TO ANY COMMAND AN EXPLANATION
OF THAT COMMAND MAY BE GIVEN
```

This menu allows you to build your molecule by defining the Cartesian coordinates interactively (`ai`) or by reading the coordinates from an external file (`a`, `aa`). The

structure can be manipulated by the commands `sub`, `m`, `name` and `del`. The command `sy` allows you to define the molecular symmetry while `desy` tries to determine automatically the symmetry group of a given molecule.

There exists a structure library which contains the Cartesian coordinates of selected molecules, e.g. CH_4. These data can be obtained by typing for example `a ! ch4` or `a ! methane`. The data files are to be found in the directory `$TURBODIR/structures`. The library can be extended.

You can perform a geometry optimization at a force field level to preoptimize the geometry. For this purpose the Universal Force Field (UFF) developed from Rappé et al. in 1992 [7] has been implemented in the `uff` module (see also Section 5.4). This can also be used to calculate an analytical approximate cartesian Hessian. If one does so, the start Hessian for the *ab initio* geometry optimization is this Hessian instead of the diagonal one (`$forceinit on carthess` for `relax` module).

Recommendation

Here is an easy way to get internal coordinates, which should work.

Have `coord` ready before calling `define`. In the main geometry menu proceed as follows to define *redundant internals*:

`a coord`	read `coord`
`desy`	determine symmetry, if you expect a higher symmetry, repeat with increased tolerance `desy 0.1` , you may go up to `desy 1.`.
`ired`	get redundant internals
`*`	quit main geometry menu

To define *internals*:

`a coord`	read `coord`
`desy`	determine symmetry
`i`	go to internal coordinate menu
`iaut`	automatic assignment of bends etc.
`q`	to quit bond analysis
`imet`	to get the metric, unnecessary internals are marked `d` now. If `#ideg = #k` in the head line you are done. Otherwise this did not work.
`<enter>`	go back to main geometry menu
`*`	quit main geometry menu

To define *cartesians*:

`a coord` read `coord`

`desy` determine symmetry

`*` quit main geometry menu

4.1.1 Description of commands

Main Geometry Menu

In the headline of this menu you can see the current number of atoms and molecular symmetry (we use an input for PH_3 as example). The commands in this menu will now be described briefly:

`sy` Definition of the Schönflies symbol of the molecular point group symmetry. If you enter only `sy`, `define` will ask you to enter the symbol, but you may also directly enter `sy c3v`. `define` will symmetrize the geometry according to the new Schönflies symbol and **will create new nuclei if necessary**. You therefore have to **take care** that you enter the correct symbol and **that your molecule is properly oriented**. All `TURBOMOLE` programs require the molecule to be in a standard orientation depending on its point group. For the groups C_n, C_{nv}, C_{nh}, D_n, D_{nh} and D_{nd} the z-axis has to be the main rotational axis, secondary (twofold) rotational axis is always the x-axis, σ_v is always the xz-plane and σ_h the xy-plane. O_h is oriented as D_{4h}. For T_d, the threefold rotational axis points in direction (1,1,1) and the z-axis is one of the twofold axes bisecting one vertex of the tetrahedron.

`desy` `desy` allows you to determine the molecular symmetry automatically. The geometry does not need to be perfectly symmetric for this command to work. If there are small deviations from some point group symmetry (as they occur in experimentally determined structures), `desy` will recognize the higher symmetry and symmetrize the molecule properly. If symmetry is lower than expected, use a larger threshold: `<eps>` up to 1.0 is possible.

`susy` `susy` leads you through the complete subgroup structure if you want to lower symmetry, e.g. to investigate Jahn–Teller distortions. The molecule is automatically reoriented if necessary.
Example: $T_d \rightarrow D_{2d} \rightarrow C_{2v} \rightarrow C_s$.

`ai` You may enter Cartesian atomic coordinates and atomic symbols interactively. After entering an atomic symbol, you will be asked for Cartesian coordinates for this type of atom until you enter `*`. If you enter `&`, the atom counter will be decremented and you may re-define the last atom (but you surely won't make mistakes, will you?). After entering `*`, `define` asks for the next atom type. Entering `&` here will allow you to re-define the last atom type and `*` to leave this mode and return to

the geometry main menu. Enter `q` as atom symbol if you want to use a dummy center without nuclear charge. Symmetry equivalent atoms are created immediately after you entered a set of coordinates.

This is a convenient tool to provide e.g. rings: exploit symmetry group D_{nh} to create an n-membered planar ring by putting an atom on the x-axis.

`a` *file* You may also read atomic coordinates (and possibly internal coordinates) from *file*, where *file* must have the same format as the data group `$coord` in file `control`.

The Cartesian coordinates and the definitions of the internal coordinates are read in free format; you only have to care for the keywords `$coord` and (optionally) `$intdef` and (important!) for the `$end` at the end of the file. The atomic symbol follows the Cartesian coordinates separated by (at least) one blank. For a description of the internal coordinate definitions refer to 4.1.2.

Entering '`!`' as first character of *file* will tell `define` to take *file* from the structure library. (The name following the '`!`' actually does not need to be a filename in this case but rather a search string referenced in the structure library contents file, see Section 4.1).

`aa` *file* same as `a`, but assumes the atomic coordinates to be in Å rather than a.u.

`sub` This command allows you to replace one atom in your molecule by another molecule. For example, if you have methane and you want to create ethane, you could just substitute one hydrogen atom by another methane molecule. The only requirement to be met by the substituted atom is that it must have exactly one bond partner. The substituting molecule must have an atom at the substituting site; in the example above it would not be appropriate to use CH_3 instead of CH_4 for substitution. Upon substitution, two atoms will be deleted and the two ones forming the new bond will be put to a standard distance. `define` will then ask you to specify a dihedral angle between the old and the new unit. It is also possible to use a part of your molecule as substituting unit, e.g. if you have some methyl groups in your molecule, you can create further ones by substitution. Some attention is required for the specification of this substituting unit, because you have to specify the atom which will be deleted upon bond formation, too. If you enter the filename from which the structure is to be read starting with '`!`', the file will be taken from the structure library (see Section 4.1). Definitions of internal coordinates will be adjusted after substitution, but no new internal coordinates are created.

`i` This command offers a submenu which contains everything related to internal coordinates. It is further described in Section 4.1.2.

`m` This command offers a submenu which allows you to manipulate the molecular geometry, i.e. to move and rotate the molecule or parts of it. It is further described in Section 4.1.3.

`frag` Here, the fragments will be defined as being used by the `jobbsse` script in order to do a calculation osing the counter-poise correction scheme. In this menu, up to three monomers can be defined, together with their charges and their symmetry. When assigning atom numbers to fragments, if x is entered instead of a number, the program will request the first and last atoms of a range. This will be useful for very large fragments.

`w` *file* The command `w` writes your molecular geometry and your internal coordinates to *file*. Afterwards you will be back in the geometry main menu. If the filename entered starts with '`!`', the structure will be written to the structure library.

`name` `name` allows you to change atomic identifiers turning, e.g. oxygen atoms into sulfur atoms. After entering the identifier to be changed (remember the double quotation marks : `"c ring"`), you will be asked to enter the new one. You can use question marks for characters not to be changed, e.g. you enter `"??ring"` to change `c chain` to `c ring`. If you do not enter eight characters, your input will be filled up with trailing blanks.

`del` The command `del` allows you to delete one or more atoms. After you entered the atomic list, `define` will show you a list of all atoms concerned and will ask you to confirm deleting these atoms. If any internal coordinate definitions exist, which rely on some of the deleted atoms, these definitions will be deleted, too.

`banal` The command `banal` allows you to perform a bonding analysis, that is, `define` will try to decide which atoms are bonded and which are not (according to a table of standard bond lengths which is included in the code of `define`). You must have performed this command before you can use the display commands `disb` (display bonding information) or `disa` (display bond angle information). The standard bond lengths (and the bonding analysis available from these) are also needed for the commands `sub` and `iaut` (see internal coordinate menu, Section 4.1.2). If you want to change the standard bond lengths (or define more bond lengths, because not for all possible combinations of elements a standard length is available) you can do that by creating your own file with the non-default values and by specifying its full pathname in file `.sys.data`. The file has the following simple format:

```
c - h  2.2
h - h  2.0
. - .  ...
```

The format of the entries is almost arbitrary: the two element symbols have to be separated by a bar, the new bond distance follows in free format (in atomic units). If the file cannot be read properly, a warning message is displayed.

`*` This command leaves this first main menu and writes all data generated so far to file. The default output file is the file you choose in the first question during your `define` session (usually `control`). Now the data groups `$coord` and `$intdef` will be written to file. After leaving this menu, you will enter the atomic attributes menu, which is described in Section 4.2.

4.1.2 Internal Coordinate Menu

```
INTERNAL COORDINATE MENU  ( #ideg=6       #k=2    #f=0    #d=0    #i=0 )

imet <a>  : PROVIDE B-MATRIX FOR ACTIVE INTERNAL COORDINATES
            (CHECK COMPLETENESS AND NUMERICAL QUALITY
            AND CHANGE REDUNDANT INTERNALS TO display)
idef      : SUB-MENU FOR INTERACTIVE DEFINITION OF INTERNAL COORDINATES
ideg <a>  : OUTPUT NUMBER OF TOT. SYMMETRIC INTERNAL DEGREES OF FREEDOM
iaut      : TRY AUTOMATIC DEFINITION OF INTERNAL COORDINATES
iman <a>  : MANIPULATE GEOMETRY BY CHANGING INTERNAL COORDINATE VALUES
imanat <i>: AS iman BUT STARTING AT INTERNAL COORD. NUMBER i
ic <i> <x>: CHANGE STATUS OF INTERNAL COORDINATE <i> TO <x>
            e.g. ic 5 d TO MAKE 5TH COORD. display OR ic k d
irem <i>  : REMOVE INTERNAL COORDINATE <i>,
            e.g. irem d TO REMOVE ALL display COORDS
dis       : ANY DISPLAY COMMAND e.g. disi OR disc
disiat <i>: AS disi BUT STARTING AT INTERNAL COORD. NUMBER i

WHERE  <a>= OPTIONAL ATOMIC SET (DEFAULT=all)
       <i>= INDEX(LIST) OF INTERNAL COORDINATE(S) LIKE 3-6,8 OR <i>=<x>
       <x>= STATUS OF INTERNAL COORDINATE = k, f, d OR i
ADDING A QUESTION MARK TO ANY COMMAND MAY PROVIDE EXPLANATIONS

ENTER COMMAND OR HIT >return< TO GET BACK TO GEOMETRY MAIN MENU
```

The parameters in the headline of this menu have the following meanings:

`#ideg` is the total number of symmetry restricted degrees of freedom.

`#k` is the number of *active* internal coordinates specified up to now. Only these coordinates are optimized during a geometry optimization.

`#f` is the number of *fixed* internal coordinates specified. These coordinates will be included in the **B**-matrix (see command `imet`), but their values will not be changed during geometry optimization.

`#d` is the number of internal coordinates whose values will only be displayed (e.g. by command `disi`), but no gradients will be calculated for these coordinates nor will they be included in the geometry optimization.

`#i` means the number of coordinates which are defined, but will be completely ignored, i.e. they are not even displayed on the screen and will not be used by any program (this is the waste-paper-basket of `define`).

Note that the `#k` plus `#f` must equal the number of degrees of freedom (`#ideg`) of your molecule, if you want to perform a geometry optimization. If you have less coordinates than degrees of freedom, you will have to specify further ones (commands `idef` or `iaut`, see below); if you have more coordinates than degrees of freedom, you will have to throw away some of them (commands `irem` or `imet`, see below).

The commands in this menu allow you to define internal coordinates for your molecule, adjust your geometry to special values of these internal coordinates and to control the numeric reliability of the chosen set of internal coordinates. In detail, the commands act as follows.

Description of commands

`imet` *a* This command computes the so-called **B**-matrix, which is the matrix of the derivatives of the (*active* and *fixed*) internal coordinates with respect to Cartesian coordinates. This matrix is used in program `relax` for the conversion from Cartesian coordinates and gradients to internal ones and vice versa. If this matrix is singular (or even nearly singular) this indicates a linear dependency of your internal coordinate set. As a consequence the geometry update (more exactly the transformation of the updated internal coordinates into Cartesian ones) will fail. This may also happen in the course of a geometry optimization if the coordinates run into linear dependency during their optimization. `imet` checks the **B**-matrix and removes linear dependent internal coordinates from your list (their status is changed from `#k` or `#f` to `#d`). If **B** is only near singular, a warning is issued and the lowest eigenvalue(s) as well as the corresponding eigenvector(s) are displayed. In this case, you should try to find better internal coordinates (although this may not always be possible). After the command `imet`, there may be too few (active plus fixed) internal coordinates, but certainly not too many (because linear dependencies have been eliminated). Perhaps you will have to add new ones or—better!—try command `iaut` or `ired` in the preceding menu.

Command `imet` should be used always after creating internal coordinates with `iaut` or `idef` (especially after `iaut`, because this command creates usually an overcomplete set of internal coordinates).

`idef` `idef` unfolds a little submenu where you can define internal coordinates manually. The exact procedure of the definition will be described below in a separate section.

`ideg` a This command gives you the number of symmetry restricted degrees of freedom (for the atomic set specified by a). Without symmetry, this is just $3N-6$, where N is the number of atoms, but if there is symmetry, some of these degrees of freedom will violate symmetry and therefore are not valid. For geometry optimizations, only the symmetry allowed degrees of freedom are needed, because the symmetry requirements are imposed anyway. In connection with the optional atomic set a this command can help you to find out, in which part of a complicated molecule internal coordinates are missing, if you fail to get the full number of `#ideg` (which equals the result of `ideg all`) for the molecule as a whole.

`iaut` `iaut` tries an automatic definition of internal coordinates. This command relies on an recursive procedure which tries to simplify the molecule as far as possible and then starts the definition of internal coordinates. At present not all molecular topologies are supported, therefore it may happen that no internal coordinates can be assigned to your molecule or at least a part of it. However, for all cases in which an automatic assignment of coordinates is possible, `iaut` has up to now proved to provide very good internal coordinates. If `iaut` works for your molecule (and in most non-pathological cases it will) we recommend strongly to use these coordinates, as they may help you to save several cycles in the geometry optimization procedure. After creating internal coordinates with `iaut` you should always use `imet` (see above), because `iaut` may provide an overcomplete set of coordinates. All coordinates which conflict with the molecular symmetry are set to *ignore* by `iaut`.

`iman` a `iman` allows you to modify the values of internal coordinates. If you specify a list of atoms a only those internal coordinates which refer to only these atoms will be handled. You will get a list of all (active and fixed) internal coordinates and their current values and you will be able to enter a new value for each of them if you like. Default (`<enter>`) keeps the value shown. Be aware that all distances are given in atomic units (1 a.u. = 52.9 pm).

`ic` i x This option allows you to change the status of a coordinate, e.g. from *active* to *display* or every other combination. The syntax is `ic 5 d`, if coordinate no. 5 is to be set to *display*, or `ic k d`, if all active coordinates are to be set to *display*.

`irem` i This option allows you to delete definitions of internal coordinates from your list. The indices of the internal coordinates always refer to the full list of coordinates including *display* and *ignore* coordinates. To make sure you delete the right ones, use `disi`. Also the indices will immediately change if you delete coordinates. If you want to delete several coordinates, this is therefore done most easily if you delete them in order of descending indices (because deletion of a coordinate has only an effect on the coordinates with higher indices). After choosing the coordinates

to be deleted, a list of all coordinates involved will be displayed and you will be asked to confirm deletion.

The syntax is simply `irem 5` to delete internal coordinate no. 5 or `irem d` to remove all 'display' coordinates.

Hitting `<return>` will bring you back to the geometry main menu.

Interactive Definition of Internal Coordinates

If you choose `idef` in the internal coordinate menu, you will get the following information:

```
ENTER INTERNAL COORDINATE DEFINITION COMMAND
                <x> <type> <indices>
WHERE    <x>       = k    f    d    i
         <type>    = stre invr bend outp tors linc linp
                     comp ring pyrm bipy pris cube octa
THESE COMMANDS WILL BE EXPLAINED IN DETAIL IF YOU ENTER
<x> <type>? FOR SOME CHOICE OF <x> AND <type>, E.G. k stre ?
DEFAULT=GO BACK TO INTERNAL MAIN MENU    DISPLAY=dis
```

The `<x>` means the status (see page 47) of the internal coordinate entered (`k`, `f`, `d`, `i`). The syntax is:

```
k stre 1 2
d tors 3 6 2 7
f bend 3 4 5
i outp 3 4 7 9
```

Note that in the third example atom 5 is the *central atom* of the angle!

Specification of available internal coordinates

The following types of coordinates are available:

`stre` The `stre` (for **stre**tch) describes a distance between two atoms. It needs only two atomic indices to be given, the order of which is arbitrary.

`invr` The `invr` coordinate (for **inv**erse **r**) describes an inverse distance. The declaration is the same as for `stre`, but in some cases (if you are far away from the minimum) the use of `invr` may result in better convergence.

`bend` `bend` describes a bond angle. It requires three atoms to be specified, of which the **third** one is the atom at the apex.

`outp` Out-of-plane angle: `outp` *abcd* is the angle between bond $a-d$ and plane $b-c-d$.

`tors` Dihedral angle: `tors` *abcd* is the angle between the planes $a-b-c$ and $b-c-d$.

`linc` This is a special coordinate type to describe the bending of a near-linear system. `linc` *abcd* describes the collinear bending of $a-b-c$ (where the angle is defined as for `bend`: the apex atom appears last) **in** the plane of $b-c-d$ (see also below, command `linp`). The system $b-c-d$ has to be non-linear, of course.

`linp` This coordinate is similar to `linc`, but describes the bending of $a-b-c$ *perpendicular* to the plane $b-c-d$. These two types of coordinates are in most cases sufficient to describe the bending of near-linear systems. An example may help you to understand these two coordinate types. Consider ketene, H_2CCO, which contains a linear system of three atoms. Without symmetry, this molecule has 9 degrees of freedom. You could choose the four bond lengths, two CCH angles and the out-of-plane angle of the C–C bond out of the CHH–plane. But then two degrees of freedom still remain, which cannot be specified using these *normal* coordinate types. You can fix these by using `linc` and `linp`. The two coordinates `linc 1 3 2 4` and `linp 1 3 2 4` (where 1=oxygen, 2=carbon, 3=carbon, 4=hydrogen) would solve the problem.

`comp` The type `comp` describes a **comp**ound coordinate, i.e. a linear combination of (primitive) internal coordinates. This is often used to prevent strong coupling between (primitive) internal coordinates and to achieve better convergence of the geometry optimization. The use of linear combinations rather than primitive coordinates is especially recommended for rings and cages (see ref. [19]). Command `iaut` uses linear combinations in most cases.

After you entered `k comp` n where n is the number of primitive internal coordinates to be combined, you will be asked to enter the type of the coordinate (`stre`, `bend`, ...). Then you will have to enter the weight (the coefficient of this primitive coordinate in the linear combination) and the atomic indices which define each coordinate. The definition of the primitive coordinates is the same as described above for the corresponding coordinate types. It is not possible to combine internal coordinates of different types.

`ring` This type helps you to define special ring coordinates. You only have to enter `k ring` n where `n` is the ring size. Then you will be asked for the atomic indices of all atoms which constitute the ring and which must be entered in the same order as they appear in the ring. The maximum number of atoms in the ring is 69 (but in most cases the ring size will be limited by the maximum number of atoms which is allowed for `define`).

Hitting `<return>` will bring you back to the internal coordinate menu where you can see the new number of internal coordinates in the headline.

4.1.3 Manipulating the Geometry

Note that the molecular geometry can be modified with the `iman` command of the internal coordinate menu described earlier, if internal coordinates has been defined. Another option is to select `m` in the geometry main menu which provides the following submenu:

```
CARTESIAN COORDINATE MANIPULATION MENU :
move   : TRANSLATE AND/OR ROTATE PART OF THE MOLECULE
inert  : MOVE MOLECULE SO THAT COORDINATE AXES BECOME
         PRINCIPAL AXES OF INERTIA
mback  : RESTORE PREVIOUS MOLECULAR GEOMETRY
dis    : DISPLAY MOLECULAR GEOMETRY
YOU MAY APPEND A QUESTION MARK TO ANY OF THESE COMMANDS
FOR FURTHER EXPLANATIONS.
HIT >return< OR USE ANY GEOMETRY COMMAND NOT IN THIS LIST
TO TERMINATE THIS MENU.
UPON TERMINATION THE MOLECULAR SYMMETRY WILL BE ENFORCED
ACCORDING TO SYMMETRY GROUP c3v .
```

The meaning of the commands `inert` and `mback` should be clear; command `move` allows you to manipulate the geometry of your molecule. After entering `move`, you will be asked to specify a set of atoms on which the command shall act. You can use this to manipulate only a part of your molecule, e.g. if you are building a structure from subunits and you want to adjust their relative arrangement. As long as you stay in this menu, the molecular symmetry needs not be correct (so that you can try different movements and/or rotations), but as soon as you leave it, the geometry will be symmetrized according to the present Schönflies symbol. After you specified the atomic set to be considered, you get the following information:

```
INPUT DIRECTION OF MOVEMENT OR LOCATION OF ROTATION AXIS
EITHER AS A COORDINATE TRIPLE SEPARATED BY BLANKS,
OR AS TWO ATOMIC INDICES SEPARATED BY KOMMA, OR x OR y OR z
OR ENTER ANY DISPLAY COMMAND FIRST OR & TO GO BACK
```

You can thus specify the direction of movement (or the rotational axis) in the form `0. 0. 1.` or simply `z` (which both describes the z-axis) or `1.3256 -3.333 0.2218` for an arbitrary axis. If you want to specify an axis which is related to your molecule, you may also enter two atomic indices which define it. After having specified the axis, you have to enter the distance of movement and the angle of rotation. If you want to perform a simple rotation, enter 0 for the distance of movement and if you want to simply move your structure, enter 0 for the rotational angle.

You can leave this menu and return to the geometry main menu by hitting `<return>` or by entering any command of the geometry main menu.

4.2 The Atomic Attributes Menu

After you specified the molecular geometry and symmetry and wrote this data to file, you will encounter the atomic attributes menu, which is the second of the four

main menus. You will enter this menu, if all necessary data cannot be read from your input file or if you do not use an input file. This menu deals with the specification of basis sets and other data related to the atom type:

```
ATOMIC ATTRIBUTE DEFINITION MENU  ( #atoms=5     #bas=5     #ecp=0     )

b    : ASSIGN ATOMIC BASIS SETS
bb   : b RESTRICTED TO BASIS SET LIBRARY
bl   : LIST ATOMIC BASIS SETS ASSIGNED
bm   : MODIFY DEFINITION OF ATOMIC BASIS SET
bp   : SWITCH BETWEEN 5d/7f AND 6d/10f
lib  : SELECT BASIS SET LIBRARY
ecp  : ASSIGN EFFECTIVE CORE POTENTIALS
ecpb : ecp RESTRICTED TO BASIS SET LIBRARY
ecpi : GENERAL INFORMATION ABOUT EFFECTIVE CORE POTENTIALS
ecpl : LIST EFFECTIVE CORE POTENTIALS ASSIGNED
ecprm: REMOVE EFFECTIVE CORE POTENTIAL(S)
c    : ASSIGN NUCLEAR CHARGES (IF DIFFERENT FROM DEFAULTS)
cem  : ASSIGN NUCLEAR CHARGES FOR EMBEDDING
m    : ASSIGN ATOMIC MASSES (IF DIFFERENT FROM DEFAULTS)
dis  : DISPLAY MOLECULAR GEOMETRY
dat  : DISPLAY ATOMIC ATTRIBUTES YET ESTABLISHED
h    : EXPLANATION OF ATTRIBUTE DEFINITION SYNTAX
*    : TERMINATE THIS SECTION AND WRITE DATA OR DATA REFERENCES TO control
GOBACK=& (TO GEOMETRY MENU !)
```

The headline gives you the number of atoms, the number of atoms to which basis sets have already been assigned and the number of atoms to which effective core potentials have already been assigned. Most of the commands in this menu deal with the specification of basis sets and pseudopotentials.

Basis sets available

The following basis sets are available on `$TURBODIR/basen/`, which you may inspect to see which other basis sets are supported automatically. The corresponding publications can be found here 1.3.

`SV(P)` or `def-SV(P)`	for routine SCF or DFT. Quality is about 6–31G*.
`TZVP` or `def-TZVP`	for accurate SCF or DFT. Quality is slightly better than 6–311G**.
`TZVPP` or `def-TZVPP`	for MP2 or close to basis set limit SCF or DFT. Comparable to 6–311G(2df).
`QZVP and QZVPP`	for highly correlated treatments; quadruple zeta + 3d2f1g or 4d2f1g (beyond Ne), 3p2d1f for H.

These basis sets are available for atoms H–Kr, and the split-valence (SV) and valence-triple-ζ (TZV) basis sets types with ECPs also for Rb–Rn, except lanthanides.

For calculations with the programs `rimp2` and `ricc2` optimized auxiliary basis sets are available for the basis sets SV(P), SVP, TZVP, TZVPP, and QZVPP.

NEW: New sets of basis functions, partly identical with those mention above, denoted def2-XYZ are available for atoms H–Rn [6]. The def2 basis sets for 5p and 6p block elements are designed for small core ECPs (ECP-28, ECP-46 and ECP-60). For each family, SV, TZV, and QZV, we offer two sets of polarisation functions leading to:

`def2-SV(P)` and `def2-SVP`

`def2-TZVP` and `def2-TZVPP`

`def2-QZVP` and `def2-QZVPP`

We strongly recommended the new def2-basis, since they have been shown to provide consistent accuracy across the periodic table.

Recommendation

Use the same basis set type for all atoms; use ECPs beyond Kr since this accounts for scalar relativistic effects.

New basis sets (def2-XYZ): MP2 implies RI-MP2 and RICC2

exploratory	MP2: SVP
almost quantitative	DFT: SV(P), HF: SVP, MP2: TZVPP; properties (HF and DFT): TZVPP
quantitative	DFT: TZVP, HF: TZVPP, MP2: QZVPP
basis set limit	DFT: QZVP, HF: QZVP

If you want a better basis than SV(P), assigned automatically, use `b all def2-TZVP` (or another basis). The assignment can be checked by `bl`.

Diffuse functions should only be added if really necessary. E.g. for small anions or treatment of excited states use: `TZVP` instead of `SVP` + *diffuse*. This is more accurate and usually faster. Only for excited states of small molecules or excited states with (a partial) Rydberg character add additional diffuse functions (e.g. by using the aug-cc-pVTZ basis) as well as the keyword `diffuse`, for more information, see page 215 in the keyword section.

[**Old basis sets** (def-XYZ): For standard correlated calculations (MP2, RI-MP2, RI-CC2) use the doubly-polarized TZVPP (or def-TZVPP) basis.]

Correlation-Consistent (Dunning) Basis Sets

Dunning basis sets like `cc-pVDZ`, `cc-pVTZ`, `cc-pVQZ` are also supported, e.g. by `b all cc-pVTZ`. But these basis sets employ generalized contractions for which

TURBOMOLE is not optimized. This has in particular strong effects on the performance of all programs which use 4-index electron repulsion integrals, for RI-MP2 and RI-CC2 this is partially compensated by the RI-approximation.

The following correlation consistent basis sets are available in the TURBOMOLE basis set library:

cc-pVXZ	standard valence X-tuple zeta basis sets (X = D, T, Q, 5, 6); available for H, He, Li–Ne, Na–Ar, K, Ca, Ga–Kr. (cc-pV6Z only for H, He, B–Ne, Al–Ar; for Al–Ar also the recommended newer cc-pV(X+d)Z sets are available)
cc-pwCVXZ	weighted core-valence X-tuple zeta basis sets (X = D, T, Q, 5); available for H, He, B–Ne, Al–Ar. (for Al–Ar also the recommended combination of the cc-pV(X+d)Z sets with the core valence functions (wC), i.e. the cc-pwCV(X+d)Z basis set are available)
aug-	diffuse functions for combination with the basis sets cc-pVXZ, cc-pV(X+d)Z, cc-pwCVXZ or cc-pV(X+d)Z; available for H, He, B–Ne, Al–Ar with X = D–6 and Ga–Kr with X = D–5.

For calculations with the programs rimp2 and ricc2 optimized auxiliary basis sets are available for the basis set series cc-pVXZ, cc-pV(X+d)Z, cc-pwCVXZ, cc-pwCV(X+d)Z, aug-cc-pVXZ, aug-cc-pV(X+d)Z, aug-cc-pwCVXZ, and aug-cc-pwCV(X+d)Z with X = D, T, Q, 5, but not for X = 6.

4.2.1 Description of the commands

`b` With `b` you can specify basis sets for all atoms in your molecule. After entering `b` you will be asked to specify the atoms to which you want to assign basis sets. You can do this in the usual ways (refer to Section 4.0.2), including `all` and `none`. Then you will be asked to enter the *nickname* of the basis set to be assigned. There are two principal ways to do this:

1) If you are in the *append* mode, the nickname you entered will be appended to the atomic symbol of the element under consideration. This is especially useful if you want to assign basis sets to different atoms with one command. For example, if you want to assign basis sets to hydrogen and oxygen atoms and you enter only `DZ`, the basis sets `h DZ` and `o DZ` will be read from the basis set library.

2) If you are in the *non-append* mode, no atomic symbol will be inserted in front of the nickname entered. Therefore you have to enter the *full* basis set nickname, e.g. `h DZ`. This mode is advantageous if you want to assign basis sets to dummy centers (i.e. points without

nuclear charge but with basis functions, e.g. for counterpoise calculations) or if you want to use the basis set nickname `none` (which means no basis functions at this atom).

You can switch between the two modes with '`+`' (switches to append mode) and '`-`' (switches to non-append mode).

Once you have specified your basis set nickname, `define` will look in the standard input file (normally `control`) for this basis set. If it can not be found there, you can switch to the standard basis set library (if you did not use a standard input file, the standard library will be searched immediately). If the basis set cannot be found there, you are asked either to enter a new standard library (which will be standard only until you leave this menu) or another input file, where the basis set can be found. If you do not know the exact nickname of your basis set, you may abbreviate it by '`?`', so you could enter `h DZ?` to obtain basis sets like `h DZ`, `h DZP`, `h DZ` *`special`*, etc. `define` will give you a list of all basis sets whose nicknames match your search string and allows you to choose among them. You may also use the command `list` to obtain a list of all basis sets cataloged.

`bb` `bb` does essentially the same as `b` but does not search your default input file for basis sets. Instead it will look in the basis set library immediately.

`bl` `bl` gives you a list of all basis sets assigned so far.

`bm` This command is used to modify basis sets which are already assigned. The corresponding submenu is self-explanatory, but we recommend to change directly the file `basis`.

`bp` The `TURBOMOLE` programs normally work with basis sets of $5d$-functions (which means they delete the s-component of the full $6d$-set). `bp` allows to switch between the proper $5d/7f$-set and the Cartesian $6d/10f$-set.

`ecp` This command allows you to specify effective core potentials for some atoms. The assignment works exactly like the specification of basis sets (see above).

`ecpb` This one does the same as command `ecp`, but restricted to the basis set library (the input file will not be used).

`ecpi` `ecpi` gives you some general information about what type of pseudopotentials is supported. For more information we refer to [20] and references therein.

`ecpl` `ecpl` gives you a list of all pseudopotentials assigned so far.

`ecprm` `ecprm` allows to remove a pseudopotential assignment from the list. This command is useful if you want to perform an all electron calculation after an ECP treatment.

`c` Command `c` assigns a special nuclear charge to an atom. This is useful to define dummy centers for counterpoise calculations where you set the nuclear charge to zero.

`m` This command allows you to assign non-default atomic masses to an atom. Use this if you want to analyze isotopic shifts of calculated harmonic frequencies. The standard masses are those of the natural isotope mix.

`dat` `dat` gives you a list of all data already specified.

`*` This is again the usual command to leave a menu and write all data to file `control` (or any other output file). It is not possible to leave this menu unless basis sets have been specified for all atoms in your molecule. If you do not want to use a basis set for one or more atoms, use the basis set nickname `none`. On leaving this menu, the data groups `$atoms` and `$basis` will be written to the output file.

After you finished this menu, you will enter the third main menu of `define` which deals with start vectors and occupation numbers.

4.3 Generating MO Start Vectors

4.3.1 The MO Start Vectors Menu

This menu serves to define the occupation numbers, and to generate the start vectors for HF-SCF and DFT calculations. They may be constructed from earlier SCF calculations (perhaps employing another basis set, type `use`), by Hamilton core guess (`hcore`), or by an extended Hückel calculation which can be performed automatically (`eht`). An occupation of the start orbitals will be proposed and can be modified if desired.

```
OCCUPATION NUMBER & MOLECULAR ORBITAL DEFINITION MENU

 CHOOSE COMMAND
 infsao     : OUTPUT SAO INFORMATION
 atb        : Switch for writing MOs in ASCII or binary format
 eht        : PROVIDE MOS && OCCUPATION NUMBERS FROM EXTENDED HUECKEL GUESS
 use <file> : SUPPLY MO INFORMATION USING DATA FROM <file>
 man        : MANUAL SPECIFICATION OF OCCUPATION NUMBERS
 hcore      : HAMILTON CORE GUESS FOR MOS
 flip       : FLIP SPIN OF A SELECTED ATOM
 &          : MOVE BACK TO THE ATOMIC ATTRIBUTES MENU
 THE COMMANDS  use  OR  eht  OR  *  OR q(uit) TERMINATE THIS MENU !!!
 FOR EXPLANATIONS APPEND A QUESTION MARK (?) TO ANY COMMAND
```

Recommendation

You will normally only need to enter `eht`. For the EHT-guess, `define` will ask for some specifications, and you should always choose the default, i.e. just `<enter>`. Of importance is only the molecular charge, specified as `0` (neutral, default), `1` or `-1` etc.

Based on the EHT orbital energies `define` proposes an occupation. If you accept you are done, if not you get the "occupation number assignment menu" explained in 4.3.2.

Description of Commands

`infsao` Command `infsao` provides information about the symmetry adapted basis which is used for the SCF-calculation. To exploit the molecular symmetry as efficiently as possible, `TURBOMOLE` programs do not use the simple basis which you specified during your `define` session. Instead it builds linear combinations of equal basis functions on different—but symmetry equivalent—atoms. This basis is then called the SAO-basis (**S**ymmetry **A**dapted **O**rbital). It has the useful property that each basis function transformed to this basis transforms belongs to one irreducible representation of the molecular point group (that is, the basis reflects the full molecular symmetry as specified by the Schönflies symbol). `infsao` gives you a listing of all symmetry adapted basis functions and their constituents either on file or on the screen. This may help you if you want to have a closer look at the SCF vectors, because the vector which is output by program `dscf` is written in terms of these SAOs.

`atb` Molecular orbitals can be written either in ASCII or in binary format. This command switches from one option to the other, and it is highly recommended to read which setting is currently active. ASCII format is portable and allows the usage of `TURBOMOLE`input files on different systems with incompatible binary format. Binary format is faster and smaller files will be written. The external program `atbandbta` can be used to transform existing `mos`, `alpha`, and `beta` files from ASCII to binary format and vice versa.

`eht` `eht` performs an extended Hückel calculation for your molecule. The orbital energies available from this calculation are then used to provide occupation numbers for your calculation and the Hückel MOs will be projected onto the space that is spanned by your basis set. This start-vectors are not as good as the ones which may be obtained by projection of an old SCF vector, but they are still better than the core Hamiltonian guess that is used if no start vectors are available. When using this command, you will be asked if you want to accept the standard Hückel parameters and to enter the molecular charge. Afterwards you will normally get a list of the few highest occupied and lowest unoccu-

pied MOs, their energies and their default occupation. If you don't want to accept the default occupation you will enter the occupation number assignment menu, which is described in Section 4.3.2. Note that the occupation based on the Hückel calculation may be unreliable if the difference of the energies of the HOMO and the LUMO is less than 0.05 a.u. (you will get a warning). You will also have to enter this menu for all open-shell cases other than doublets.

`use` *file* With command `use` you are able to use information about occupied MOs and start vectors from a former calculation on the same molecule. *file* should be the path and name of the `control` file of this former calculation, of which all data groups related to occupation numbers and vectors will be read. As the new generated data will overwrite the existing data if both resist in the same directory, it is best and in some cases necessary to have the data of the former calculation in another directory than the one you started the `define` session in. Then just type `use <path>/control` to construct a new SCF vector from the data of the old calculation, without changing the old data. The data groups `$closed shells` and `$open shells` will be taken for your new calculation and the SCF vector from the old calculation will be projected onto the space which is spanned by your present basis set. These start vectors are usually better than the ones you could obtain by an extended Hückel calculation.

`man` `man` allows you to declare occupation numbers or change a previous declaration manually. After selecting this command, you will get a short information about the current occupation numbers:

```
------------------------------------------------------------
actual closed shell orbital selection         range
------------------------------------------------------------
 a1                                           #   1-  18
 a2                                           #   1-   1
 e                                            #   1-  13
------------------------------------------------------------

any further closed-shell orbitals to declare ? DEFAULT(y)
```

If you answer this question with `y`, you enter the orbital specification menu which will be described in Section 4.3.3.

The same procedure applies to the open-shell occupation numbers after you finished the closed-shell occupations.

`hcore` `hcore` tells programs `dscf` and `ridft` to run without a start vector (it writes the data group `$scfmo none` to file `control`). `dscf` or `ridft` will then start from the core Hamiltonian start vector, which is the vector obtained by diagonalizing the one-electron Hamiltonian. Note that you still have to specify the occupation numbers. This concerns only the

first SCF run, however, as for the following calculations the converged vector of the previous iteration will be taken. A SCF calculation with a core Hamiltonian start vector typically will take $2-3$ iterations more than a calculation with an extended Hückel start vector (a calculation with the converged SCF vector of a previous calculation will need even less iterations, depending on how large the difference in the geometry between the two calculations is).

`flip` flipping of spins at a selected atom. Requirements: converged UHF molecular orbitals and no symmetry (C_1). `define`will localize the orbitals, assign them to the atoms and give the user the possibility to choose atoms at which alpha-orbitals are moved to beta orbitals, or vice versa. This is useful for spin-broken start orbitals, but not for spatial symmetry breaking.

`*` This command (as well as `use` and `eht`) terminates this menu, but without providing a start vector. If the keyword `$scfmo` exists in your input file, it will be kept unchanged (i.e. the old vector will be taken), otherwise `$scfmo none` will be inserted into your output file, which forces a calculation without start vector to be performed. When you leave this menu, the data groups `$closed shells`, `$open shells` (optionally) and `$scfmo` will be written to file. You will then reach the last of the four main menus (the General Menu) which is described in Section 4.4.

4.3.2 Assignment of Occupation Numbers

If an automatic assignment of occupation numbers is not possible or you do not except the occupation numbers generated by the EHT, you enter the following menu:

```
OCCUPATION NUMBER ASSIGNMENT MENU ( #e=60 #c=0 #o=0)

 s         : CHOOSE UHF SINGLET OCCUPATION
 t         : CHOOSE UHF TRIPLET OCCUPATION
 u <int>   : CHOOSE UHF WITH <int> UNPAIRED ELECTRONS
 l <list>  : PRINT MO'S FROM EHT IN <list>, (DEFAULT=ALL)
 p <index> : PRINT MO-COEFFICIENTS OF SHELL <index>
 c <list>  : CHOOSE SHELLS IN <list> TO BECOME CLOSED SHELLS
 o <index> : CHOOSE SHELL <index> TO BECOME AN RHF OPEN SHELL
 a <list>  : CHOOSE SHELLS IN <list> TO BECOME UHF ALPHA SHELLS
 b <list>  : CHOOSE SHELLS IN <list> TO BECOME UHF BETA SHELLS
 v <list>  : CHOOSE SHELLS IN <list> TO BECOME EMPTY SHELLS
 &         : REPEAT THE EXTENDED HUECKEL CALCULATION
 *         : SAVE OCCUPATION NUMBERS & GO TO NEXT ITEM
 dis       : GEOMETRY DISPLAY COMMANDS
 e         : CALCULATE EHT-ENERGY
 f         : FURTHER ADVICE
 <int>     = INTEGER
 <index>   = INDEX OF MO-SHELL ACCORDING TO COMMAND s
 <list>    = LIST OF MO-SHELL INDICES (LIKE  1-5,7-8,11)
```

Recommendation

Enter `l` to get a list of eht MO energies. Then make up your mind on what to do: closed shell, RHF open shell (not allowed for DFT) or UHF. Look at the examples below.

RHF `c 1-41,43,45` to define these levels to be doubly occupied.

UHF `a 1-5 alpha` levels to be occupied, `b 1-3,5 beta` levels to be occupied. Or simply, `s`, `t`, or `u 1` to get singlet, triplet or doublet occupation pattern.

ROHF `c 1-41,43,45` levels to be doubly occupied; `o 42` level 42 should be partially occupied. You will then be asked to specify the occupation. If there are more open shells you have to repeat, since only a single open shell can be specified at a time. Watch the headline of the menu, which tells you the number of electrons assigned to MOs.

Description of Commands

`s` *list* This command gives you a listing of all MOs and their energies as obtained from the extended Hückel calculation. For NH_3 in C_{3v} and `TZVP` you get, e.g.:

```
ORBITAL SYMMETRY  ENERGY      SHELL   CUMULATED  CL.SHL OCC.  OP.SHL OCC.
(SHELL)   TYPE             DEGENERACY SHELL DEG. PER ORBITAL  PER ORBITAL
    1    1a1      -15.63244      2           2      0.0000      0.0000
    2    2a1       -0.99808      2           4      0.0000      0.0000
    3    1e        -0.64406      4           8      0.0000      0.0000
    4    3a1       -0.57085      2          10      0.0000      0.0000
    5    2e         0.30375      4          14      0.0000      0.0000
    6    4a1        0.87046      2          16      0.0000      0.0000
TO CONTINUE, ENTER <return>
```

`p` *index* This allows you to get the linear combination of basis functions which form the MO-index. Note that this refers *not* to the basis set you specified, but to the extended Hückel basis. *index* must be a single index, not an index list.

`c` *list* This command allows you to specify closed shells. Their occupation will be 2 per MO, the total occupation the shell degeneracy which you can obtain by using command `s`. *list* is a list of shell indices like `1-13` or `1,3-5,7`.

`o` *index* This command allows you to specify open shells. *index* must be a single shell index, not an index list. You will then be asked for the number of electrons **per MO** which shall be contained in this shell. For example, for a fluorine atom you should choose `o n` (where `n` is the index of the *p*-shell) and an occupation of 5/3 per MO. You may enter the

occupation numbers as simple integers or as integer fractions, e.g. `1` for the s-occupation in sodium, `5/3` for the p-occupation in fluorine.

`v` *list* With this command you can remove an orbital occupation, if you specified a wrong one. *list* is again a list of shell indices in usual syntax.

`&` This command has a different meaning in this menu than in the rest of `define`. Here it will repeat the extended Hückel calculation (perhaps you want to change some Hückel parameters for the next one).

`*` `*` will *not* bring you back to the occupation numbers menu, but will terminate the whole occupation number and start vector section and will bring you to the last main menu, which is described in Section 4.4. If you want to leave this menu without assigning all electrons in your molecule to shells, `define` will issue a warning and suggest to continue defining occupation numbers. You can ignore this warning, if you do not want to assign all electrons.

`e` Calculates and displays the extended Hückel total energy of your molecule.

`f` `f` will give you some information about the commands in this menu.

You may overwrite occupation numbers once given by just redefining the corresponding shell. For example, if you choose shells 1–10 as closed shells and afterwards shell no. 9 as open shell (with any occupation number), the open shell will be correctly assigned.

4.3.3 Orbital Specification Menu

`define` provides the possibility to assign the occupation numbers of the MOs manually, if you like. To do that, use the command `man` in the occupation number main menu and you will arrive at the following submenu:

```
------------- ORBITAL SPECIFICATION MENU --------------

<label> <list>  : select orbitals within <list>
-<label> <list> : skip orbitals within <list>
&               : ignore input for last label
clear           : clear all assignments
p(rint)         : print actual orbital selection
for help, type ? or help // for quit, type * or q(uit)
```

Depending on whether you are in the closed- or in the open-shell section, the commands of this menu refer only to the corresponding type of orbitals. The commands of this menu do not need much explanation. `<label>` is the irrep label of one irreducible representation of the molecular point group (e.g. `a1`, `b2`, `t1g`, ...). `<list>` is a list of orbital indices within this *irrep* (e.g. `1,2,4` or `1-8,10,11`). `p` or `print` will give you the same listing of the orbital occupations as you saw before entering this

menu. After you leave this submenu, you will be back in the occupation numbers main menu.

4.3.4 Roothaan Parameters

In open-shell calculations within the restricted Hartree–Fock ansatz (ROHF), the coupling between the closed and the open shells must be specified using two parameters `a` and `b`, which depend on the type of the open shell, the number of electrons in it (the electron configuration), but also on the state to be calculated. For example, there are three states arising from the $s^2\,p^2$ configuration of an atom (^{3}P, ^{1}D, ^{1}S) which have different values of a and b. For the definition of these parameters and their use refer to Roothaan's original paper [21]. For simple cases, `define` sets these parameters automatically. If not, you have to enter them yourself. In this case, you will get the following message:

```
ROOTHAAN PARAMETERS a AND b COULD NOT BE PROVIDED ...
TYPE IN ROOTHAAN a AND b AS INTEGER FRACTIONS
OR ENTER val FOR AN AVERAGE OF STATES CALCULATION
OR ENTER  &  TO REPEAT OCCUPATION NUMBER ASSIGNMENT
```

Note that *not* all open shell systems can be handled in this way. It is possible to specify a and b for atomic calculations with s^n, p^n, d^1, and d^9 configurations and for calculations on linear molecules with π^n and δ^n configurations. Furthermore, it is possible to do calculations on systems with half-filled shells (where a=1, b=2). In the literature you may find tabulated values for individual states arising from d^n configurations, but these are **not** correct. Instead, these are parameters for an average of all states arising from these configurations. You can obtain these values if you enter `val` on the above question. For a detailed description see Section 6.3.

4.3.5 Start-MOs for broken symmetry treatments ("flip")

Broken-symmetry treatments suggested by e.g. Noodleman or Ruiz are a popular tool for the calculation of spin coupling parameters in the framework of DFT. As an example one might consider two coupled Cu^{II} centers, e.g. for a (hypothetical) arrangement like this:

```
$coord
 0.0  2.7 0.0 cu
 0.0 -2.7 0.0 cu
 0.0 -6.1 0.0 f
 0.0  6.1 0.0 f
 2.4  0.0 0.0 f
-2.4  0.0 0.0 f
$end
```

The high-spin case, a doublet with an excess alpha electron at each Cu atom, "aa" in an obvious notation, preserves D_{2h} symmetry, while the low spin state "ba" does

not. For a broken-symmetry treatment it is advidsable to calculate the high-spin state first, and then broken-symmetry state(s); from the energy difference(s) one may calculate approximate values for the spin-spin coupling parameters as described by e.g. the above authors. Access to broken-symmetry states usually is possible by the choice of appropriate start-MOs, followed by an SCF-procedure. Start MOs may be obtained by first applying a localization procedure to the MOs of the high-spin state and then by "moving" localized alpha orbitals to the beta subset.

The preparation of broken-symmetry start-MOs can be done with `define` (semi-)automatically. Prerequisite is a converged wave function for the high-spin state in C_1-symmetry, that fulfills the *aufbau* principle.

If in this case one enters `flip` in the orbital definition menu, `define` selects the occupied valence orbitals of the system (by an orbital energy criterion, which one can usually accept, unless the system is highly charged and the orbital energies are shifted). Next a Boys orbital localization procedure is carried out, which - depending on the size of the problem - may take some time. Then the user is asked:

```
ENTER INDICES OF ATOMS OR ELEMENT TO BE MANIPULATED (example:  1,2-3 or
"mn")
```

In case of our above example one may enter `"cu"`, which immediately leads to the following output (a def-SV(P) basis and the B-P functional were used for the high-spin state):

```
 RELEVANT LMOS FOR ATOM   1 cu
 ALPHA:
index occupation "energy"     s    p    d    f   (dxx  dyy  dzz  dxy  dxz  dyz)
  15    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  18    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  20    1.000     -0.335     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  22    1.000     -0.333     0.01 0.00 0.99       0.13 0.03 0.32 0.00 0.00 0.51
  23    1.000     -0.333     0.01 0.00 0.99       0.14 0.03 0.34 0.00 0.00 0.49

 BETA:
  39    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  41    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  43    1.000     -0.321     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  46    1.001     -0.318     0.05 0.00 0.95       0.00 0.43 0.51 0.00 0.00 0.00

 RELEVANT LMOS FOR ATOM   2 cu
 ALPHA:
index occupation "energy"     s    p    d    f   (dxx  dyy  dzz  dxy  dxz  dyz)
  16    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  17    1.000     -0.357     0.01 0.00 0.98       0.20 0.27 0.01 0.50 0.00 0.00
  19    1.000     -0.335     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
  21    1.000     -0.333     0.01 0.00 0.99       0.13 0.03 0.32 0.00 0.00 0.51
  24    1.000     -0.333     0.01 0.00 0.99       0.14 0.03 0.34 0.00 0.00 0.49

 BETA:
  40    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
  42    1.000     -0.326     0.00 0.00 1.00       0.33 0.08 0.09 0.00 0.00 0.50
```

```
 44    1.000     -0.321     0.00 0.00 1.00       0.00 0.00 0.00 0.00 1.00 0.00
 45    1.001     -0.318     0.05 0.00 0.95       0.00 0.43 0.51 0.00 0.00 0.00

a2b    : FLIPPING ALPHA TO BETA (default)
b2a    : FLIPPING BETA TO ALPHA
r      : repeat atom choice
```

As evident from the second column, for each Cu atom five localized alpha and four localized beta orbitals were generated which are of d-type (the sixth column labelled "d" shows values close to 1, the other columns such close to 0). The six columns at the right show the individual contributions of the six cartesian d-functions.

What has to be done to generate start MOs for the "ba"-case? Obviously one of the five localized alpha spin orbitals from the first Cu atom (atom label `1 cu`) has to become a beta spin orbital. These five orbitals have the indices `15, 18, 20, 22, 23`. In order to avoid linear dependencies, it is advisable to take the orbital that has no beta-analogue. This can be found by comparing the contributions of the six d-functions. In the present example this is the case for the localized alpha orbitals `15` and `18`: in contrast to all localized beta orbitals they show significant contributions from d_{xy}. One thus enters

```
a2b
15
```

and after confirming the replacement of original MOs with the generated start-MOs one is finally asked

```
 It is advisable to modify damping and orbital shift in the following way:

 $scfdamp   start=5.000  step=0.050  min=0.500
 $scforbitalshift  automatic=1.0
 $scfiterlimit 999

 Do you want to replace the corresponding entries in the control-file? (y)
```

which should be confirmed, as otherwise the prepared spin state might be destroyed during the SCF iterations.

From this input one may start the SCF(HF/DFT)-procedure. For recommended choices of DFT functionals and formulae to calculate the coupling parameters from these energy differences please consult the papers of the above-mentioned authors. For reasons of economy, a pre-optimization by a pure (non-hybrid) DFT-functional is reasonable.

Important: For the converged wave function one should check, whether the resulting state is really the desired one. This can quite reliably be done by a Mulliken population analysis. For this purpose, add `$pop` to the control file, type `ridft -proper` or `dscf -proper`, respectively, and check the signs of the calculated numbers of unpaired electrons in the output.

4.4 The General Options Menu

After you specified all data concerning the molecule you want to examine, you are on your way to the last of the four main menus. Before reaching it, you will perhaps get a message like the following:

```
DO YOU WANT TO DELETE DATA GROUPS LIKE
    $energy
    $grad
    $hessian
    $hessian (projected)
    $last energy change
    $maximum norm of internal gradient
    $dipgrad
    $vibrational normal modes
    $vibrational spectrum
    $cartesianforce interspace
LEFT OVER FROM PREVIOUS CALCULATIONS ? DEFAULT(n)
```

`define` has scanned your input file for this session and found some data groups which might have become obsolete. If they are still acceptable depends on the changes you made during your present `define` session. They are obviously incorrect if you changed the molecule under consideration; but any change in the basis sets or the occupation numbers will make them dangerous, too, because you might not know some day if they really refer to the basis set which is defined in this `control` file. As a rough guide, delete them whenever you have made changes in one of the first three main menus during your `define` session.

After that you will reach the last main menu of `define` which helps you to control the actions of all `TURBOMOLE` programs. The meanings of the various options are explained in more detail in the description of the individual programs, therefore only a short explanation will be given here.

Now have a look at the menu:

```
 GENERAL MENU : SELECT YOUR TOPIC
 scf    : SELECT NON-DEFAULT SCF PARAMETER
 mp2/cc2: OPTIONS AND DATA GROUPS FOR MP2, CC2, ETC.
 ex     : EXCITED STATE AND RESPONSE OPTIONS
 prop   : SELECT TOOLS FOR SCF-ORBITAL ANALYSIS
 drv    : SELECT NON-DEFAULT INPUT PARAMETER FOR EVALUATION
          OF ANALYTICAL ENERGY DERIVATIVES
          (GRADIENTS, FORCE CONSTANTS)
 rex    : SELECT OPTIONS FOR GEOMETRY UPDATES USING RELAX
 stp    : SELECT NON-DEFAULT STRUCTURE OPTIMIZATION PARAMETER
 e      : DEFINE EXTERNAL ELECTROSTATIC FIELD
 dft    : DFT Parameters
 ri     : RI Parameters
 rijk   : RI-JK-HF Parameters
 trunc  : USE TRUNCATED AUXBASIS DURING ITERATIONS
 marij  : MULTIPOLE ACCELERATED RI-J
```

```
dis    : DISPLAY MOLECULAR GEOMETRY
list   : LIST OF CONTROL FILE
&      : GO BACK TO OCCUPATION/ORBITAL ASSIGNMENT MENU
* or q : END OF DEFINE SESSION
```

This menu serves very different purposes. The next subsection deals with commands required to activate and/or specify specific methods of calculation. The subsequent subsection describes commands used to select non-default options. Standard SCF calculations do not require special action, just leave the menu. The final subsection describes the settings for property calculations.

4.4.1 Important commands

DFT calculations

Command `dft` leads you to the menu:

```
STATUS OF DFT_OPTIONS:
 DFT is NOT used
   functional b-p
     gridsize m3

 ENTER DFT-OPTION TO BE MODIFIED

 func: TO CHANGE TYPE OF FUNCTIONAL
 grid: TO CHANGE GRIDSIZE
  on:   TO SWITCH ON  DFT
 Just <ENTER>, q or '*' terminate this menu.
```

To activate DFT input `on` and then specify the grid for the quadrature of exchange-correlation terms. `TURBOMOLE` offers grids `1` (coarse) to `7` (finest), and the multiple grids `m3` to `m5` [4]. The latter employ a coarser grid during SCF iterations, and grid `3` to grid `5` in the final SCF iteration and the gradient evaluation. Default is grid `m3`, for clusters with more than 50 atoms use `m4`.

The functionals supported are obtained with the command `func`:

```
SURVEY OF AVAILABLE EXCHANGE-CORRELATION ENERGY FUNCTIONALS

FUNCTIONAL      | TYPE | EXCHANGE       | CORRELATION    | REFERENCES
-------------------------------------------------------------------
slater-dirac-   | LDA  | S              |                | 1,2
exchange        |      |                |                |
s-vwn           | LDA  | S              | VWN(V)         | 1-3
vwn             | LDA  |                | VWN(V)         | 3
s-vwn_Gaussian  | LDA  | S              | VWN(III)       | 1-3
pwlda           | LDA  | S              | PW             | 1,2,4
becke-exchange  | GGA  | S+B88          |                | 1,2,5
b-lyp           | GGA  | S+B88          | LYP            | 1,2,6
b-vwn           | GGA  | S+B88          | VWN(V)         | 1-3,5
lyp             | GGA  |                | LYP            | 6
b-p             | GGA  | S+B88          | VWN(V)+P86     | 1-3,5,7
pbe             | GGA  | S+PBE(X)       | PW+PBE(C)      | 1,2,4,8
tpss            | MGGA | S+TPSS(X)      | PW+TPSS(C)     | 1,2,4,14
bh-lyp          | HYB  | 0.5(S+B88)     | LYP            | 1,2,5,6,9
                |      | +0.5HF         |                |
b3-lyp          | HYB  | 0.8S+0.72B88   | 0.19VWN(V)     | 1-3,5,6,10
                |      | +0.2HF         | +0.81LYP       |
b3-lyp_Gaussian | HYB  | 0.8S+0.72B88   | 0.19VWN(III)   | 1-3,5,6,10
                |      | +0.2HF         | +0.81LYP       |
pbe0            | HYB  | 0.75(S+PBE(X)) | PW+PBE(C)      | 1,2,4,8,11
                |      | +0.25HF        |                |
tpssh           | HYB  | 0.9(S+TPSS(X)) | PW+TPSS(C)     | 1,2,4,14,15
                |      | +0.1HF         |                |
lhf             | EXX  | EXX            |                | 12,13
b97-d           | GGA  | B97 refit      | B97 refit      | 16
b2-plyp         | DHYB |0.47(SB88)+0.53HF|0.73LYP+0.27PT2| 17
```

Default is `b-p`, i.e. B-P86, which is probably best for the whole of Chemistry [22]. For main group compounds we recommend `b3-lyp`; note that GAUSSIAN uses partly different implementations [22].

The programs `dscf` and `grad` are used to carry out conventional DFT treatments, i.e. J and K are evaluated without approximations.

RI-J calculations

For non-hybrid functionals we strongly recommend the RI-J procedure, which speeds up calculations by a factor 10 at least (as compared to conventional treatments) without sacrificing accuracy. Command `ri` gives:

```
STATUS OF RI-OPTIONS:
    RI IS NOT USED
    Memory for RI:           200 MB
    Filename for auxbasis: auxbasis

 ENTER RI-OPTION TO BE MODIFIED
      m: CHANGE MEMORY FOR RI
```

```
     f: CHANGE FILENAME
   jbas: ASSIGN AUXILIARY RI-J BASIS SETS
     on: TO SWITCH ON  RI
Use <ENTER>, q, end, or * to leave this menu
```

Activate RI-J with `on`, and choose with `m` the memory you can dedicate to store three-center integrals (Keyword: `$ricore`), default is 200 MB. The *more* memory, the *faster* the calculation.

A rough guide: put `$ricore` to about 2/3 of the memory of the computer. Use OS specific commands (`top` on most UNIX systems), during an `ridft` run to find the actual memory usage and then adjust `$ricore`, the keyword in `control` specifying memory.

If the option `jbas` is selected, `define` enters a submenu which allows the assignment of auxiliary basis sets (for an explanation of the menu items see Section 4.2). Where available, the program will select by default the auxiliary basis sets optimized for the orbital basis used. Please note that treatment of systems with diffuse wavefunctions may also require an extension of the auxiliary basis. For this cases enlarge the sets of s- and p-functions with diffuse functions.

The RI-J option is only supported by programs `ridft` and `rdgrad`, if you use `jobex` to optimize molecular geometry, put: `nohup jobex -ri ...`

MARI-J option

RI-J calculations can be done even more efficiently with the **M**ultipole **A**ccelerated **RI**-J (MARI-J) option, especially for larger molecules where almost linear scaling is achieved [23].

```
Parameters:
 1) precision parameter:               1.00E-06
 2) maximum multipole l-moment:              10
 3) maximum number of bins:                   8
 4) minimum separation of bins:            0.00
 5) maximum allowed extension:            20.00
 6) threshold for multipole neglect:   1.00E-18

Enter the number to change a value or <return> to accept all.
```

Just rely on the defaults.

Multiple auxiliary basis sets

With the command `trunc` you can switch on this option. Effect: a reduced auxiliary (or fitting) basis to represent the electron density is employed during SCF iterations, the final SCF iteration and the gradient are computed with the full auxiliary basis.

```
truncated RI ALREADY SWITCHED ON
DO YOU WANT TO SWITCH OFF truncation ? (default=no)
```

Note: `trunc` is presently not compatible with marij!

RI in SCF calculations

Considerable savings in CPU times are achieved with the RI technique for both Coulomb J and exchange K terms in SCF calculations, the RI-JK method [24], provided large basis sets are employed, e.g. `TZVPP`, `cc-pVTZ`, or `cc-pVQZ`. With `rijk` you get:

```
STATUS OF RI-OPTIONS:
    RI IS NOT USED
    Memory for RI:           200 MB
    Filename for auxbasis: auxbasis

 ENTER RI-OPTION TO BE MODIFIED
      m: CHANGE MEMORY FOR RI
      f: CHANGE FILENAME
   jkbas: ASSIGN AUXILIARY RI-JK BASIS SETS
     on: TO SWITCH ON  RI
 Use <ENTER>, q, end, or * to leave this menu
```

For an explanation of the menu items see Section 4.4.1. RI-JK calculations can be carried out with the program `ridft`.

Optimization to minima and transition structures using STATPT

Structure optimizations can be carried out by the program `statpt`. For minimizations no additional keywords are required. The default values are assumed, which work in most of the cases. Structure optimization is performed in internal coordinates if they have been set. Otherwise, Cartesian coordinates are used. One can switch the optimization in internal coordinates on or off, respectively in internal redundant or cartesian coordinates. For transition structure optimizations the index of transition vector has to be set to an integer value > 0 (0 means structure minimization). The value of the index specifies transition vector to follow during the saddle point search. Note, that Hessian eigenpairs are stored in ascending order of the eigenvalues, i.e. the eigenpair with the smallest eigenvector has the index 1.

The command `stp` gives:

```
 ------------------------------------------------------------------------
 CONVERGENCE CRITERIA:

 thre  1.000000E-06     thre : threshold for ENERGY CHANGE
 thrd  1.000000E-03     thrd : threshold for MAX. DISPL. ELEMENT
 thrg  1.000000E-03     thrg : threshold for MAX. GRAD. ELEMENT
 rmsd  5.000000E-04     rmsd : threshold for RMS OF DISPL.
 rmsg  5.000000E-04     rmsg : threshold for RMS OF GRAD.
```

```
 defl : set default values.

 ------------------------------------------------------------------------

 OPTIMIZATION refers to

 int   off         int: INTERNAL coordinates
 rdn   off         rdn: REDUNDANT INTERNAL coordinates
 crt   on          crt: CARTESIAN coordinates
 NOTE : options int and crt exclude each other

 ENTER STATPT-OPTIONS TO BE MODIFIED

 itvc    0             itvc : change INDEX OF TRANSITION VECTOR
 updte   bfgs          updte: change method of HESSIAN UPDATE
 hsfrq   0             hsfrq: frequency of HESSIAN CALCULATION
 kptm    0             kptm : FREEZING transition vector INDEX
 hdiag 5.000000E-01    hdiag: change DIAGONAL HESSIAN ELEMENTS
 rmax  3.000000E-01    rmax : change MAX. TRUST RADIUS
 rmin  1.000000E-04    rmin : change MIN. TRUST RADIUS
 trad  3.000000E-01    trad : change TRUST RADIUS
 ------------------------------------------------------------------------
 Just <ENTER>, q or '*' terminate this menu.
```

Excited states, frequency-dependent properties, and stability analysis

Excited state calculations with RPA or CIS (based on HF-SCF) and TDDFT procedures as well as stability analyses (SCF or DFT) are carried out by the program `escf`.

You will need a well converged HF-SCF or DFT calculation that were converged to at least `$scfconv=7`, see Section 4.4.2.

Details of calculations are specified with the command `ex`:

```
MAIN MENU FOR RESPONSE CALCULATIONS

OPTION | STATUS | DESCRIPTION
------------------------------------------------------------------------
rpas   | off    | RPA SINGLET EXCITATIONS (TDHF OR TDDFT)
ciss   | off    | TDA SINGLET EXCITATIONS (CI SINGLES)
rpat   | off    | RPA TRIPLET EXCITATIONS (TDHF OR TDDFT)
cist   | off    | TDA TRIPLET EXCITATIONS (CI SINGLES)
polly  | off    | STATIC POLARIZABILITY
dynpol | off    | DYNAMIC POLARIZABILITY
single | off    | SINGLET STABILITY ANALYSIS
triple | off    | TRIPLET STABILITY ANALYSIS
nonrel | off    | NON-REAL STABILITY ANALYSIS

ENTER <OPTION> TO SWITCH ON/OFF OPTION, * OR q TO QUIT
```

If you have selected an option, e.g. `rpas`, and quit this menu, you will get another menu:

```
SELECT IRREP AND NUMBER OF STATES
ENTER ? FOR HELP, * OR Q TO QUIT, & TO GO BACK
```

This should be self-evident.

MP2 and RI-MP2

We recommend to use MP2 together with the RI technique: program `rimp2` or `ricc2`. This is more efficient and supports the frozen core option in the gradient calculation.

The entry `mp2` leads to a submenu which allows to set some keywords for MP2 and RI-MP2 calculations, e.g. defining frozen orbitals, maximum memory usage, or assign auxiliary basis sets for RI-MP2 calculations, etc. If you want to use `ricc2`, you have to use the entry `cc2` and the submenu `ricc2` in order to assign MP2 as wavefunction model. It covers all keywords required for `rimp2` calculations, Mandatory for `rimp2` runs is the specification of the auxiliary basis set using the menu entry `cbas`. (Alternatively, the `rimp2prep` tool can be used to set the keywords needed for `rimp2` calculations.)

Conventional MP2 calculations with `mpgrad` require a number of additional settings for which it is recommended to invoke the interactive tool `mp2prep`. For geometry optimizations with `jobex` use `nohup jobex -level mp2 -ri ...`

CC2 calculations

The entry `cc2` leads to a submenu which allows to set a number of keywords essential for calculations with the program `ricc2`. In particular it allows the assignment of auxiliary basis sets (mandatory for `ricc2`!), the specification of frozen orbitals, and the definition of a scratch directory and of the maximum core memory usage.

2nd analytical derivatives

The program `aoforce` computes force constants and IR and Raman Spectra on SCF and DFT level. Analytical second derivative calculations can directly be started from converged SCF or DFT calculations. Note, that the basis is restricted to d-functions, and ROHF as well as broken occupation numbers are not allowed. For better efficiency, in case of larger systems, use the keyword `$maxcor` as described in Chapter 10 to reduce computational cost. RI will be used if the RI option for DFT has been specified.

4.4.2 Special adjustments

Adjustments described by the following menus are often better done directly in the `control` file; have a look at the keywords in Chapter 14. For common calculations just start with the defaults, and change keywords directly in `control` if you encounter problems with your calculation.

SCF options

```
ENTER SCF-OPTION TO BE MODIFIED

conv : ACCURACY OF SCF-ENERGY           $scfconv
thi  : INTEGRAL STORAGE CRITERIA        $thize  $thime
ints : INTEGRAL STORAGE ALLOCATION      $scfintunit
iter : MAXIMUM NUMBER OF ITERATIONS     $scfiterlimit
diis : DIIS CONVERGENCE ACCELERATION    $scfdiis
damp : OPTIONS FOR DAMPING              $scfdamp
shift: SHIFTING OF ORBITALS             $scforbitalshift
order: ORDERING OF ORBITALS             $scforbitalorder
fermi: THERMAL SMEARING OF OCC. NUMBERS $fermi
```

By the command `$fermi` you can switch on *smearing* of occupation numbers, and thus automatically optimize occupations and spin.

Menu drv

The most important of the derivative menus is the first one which tells the programs which derivatives to calculate. This is only necessary for special purposes and you should better not change default options.

```
------------------------------------------------------------------------
 derivative data groups '$drvopt, $drvtol'
------------------------------------------------------------------------
 option | status  | description :
------------------------------------------------------------------------
 crt    |    T    | CARTESIAN 1st derivatives
 sec    |    T    | CARTESIAN 2nd derivatives
 bas    |    F    | energy derivatives with respect to
        |         |  BASIS SET exponents/scaling factors/
        |         |  contraction coefficients
 glb    |    F    | energy derivative with respect to
        |         |  a GLOBAL scaling factor
 dip    |    T    | cartesian 1st derivatives of DIPOLE MOMENT
 pol    |    T    | nuclear contribution to POLARIZABILITY
 fa     |    F    | SPECTROSCOPIC ANALYSIS only
 tol     0.100D-06  derivative integral cutoff
------------------------------------------------------------------------
 use <opt> for enabling, -<opt> for disabling of logical switches
 <&> will bring you back to GENERAL MENU without more changes
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

The handling of these options is very simple. With the exception of `tol`, all are logical switches which are either true (or on, active) or false (or off, inactive). You can switch between the two states if you enter, for example, `crt` (to switch calculation of Cartesian first derivatives on) or `-crt` (to switch it off). The options `crt`, `sec` and `bas` should provide no problems. `glb` refers to a global scaling factor for all basis set exponents. Imagine that you would like to replace your basis set, which contains basis functions

$$\chi_\mu = (x - x_0)^l (y - y_0)^m (z - z_0)^n \exp\left[-\eta_\mu (r - r_0)^2\right]$$

by another basis set which contains basis functions

$$\chi_\mu = (x - x_0)^l (y - y_0)^m (z - z_0)^n \exp\left[-\alpha\eta_\mu (r - r_0)^2\right]$$

where α is the same for all primitive basis functions χ_μ. With command `glb` you are able to calculate analytical derivatives of the total energy with respect to α and can thus easily determine the optimum α.

`dip` enables you to calculate the first derivatives of the electric dipole moment with respect to nuclear displacements which gives you infrared intensities. `pol` allows you to calculate the contribution of the nuclear rearrangement on the electric polarizability. `fa` finally performs only a frequency analysis which means that `aoforce` will read the force constant matrix (`$hessian` or `$hessian (projected)`), diagonalize it and give you the frequencies and normal modes. `tol` is not a logical switch as the other options in this menu, but a cutoff threshold for the derivative integrals, i.e. integrals below this threshold will be neglected in the derivative calculations.

Entering `*` will bring you to the second derivative submenu.

Debug Options for the Derivative Programs

The following menu deals only with some debug options for `grad`. Use them with caution, each of them can produce lots of useless output:

```
------------------------------------------------------------------------
 derivative debug options '$drvdebug'
------------------------------------------------------------------------
 option  |status| description :
------------------------------------------------------------------------
 disp1e  |   F   | display 1e contributions to desired derivatives
 only1e  |   F   | calculate 1e contributions to desired derivatives only
 debug1e |   F   | display 1e shell contributions to desired derivatives
         |       |  (WARNING : this produces large outputs!)
 debug2e |   F   | display 2e shell contributions to desired derivatives
         |       |  (WARNING : this produces VERY large  outputs!)
 debugvib|   F   | debug switch for vibrational analysis (force only)
 notrans |   F   | disable transfer relations (gradient only!)
 novirial|   F   | disable virial scaling invariance in basis set
         |       |  optimizations (gradient only)
------------------------------------------------------------------------
 use <opt> for enabling, -<opt> for disabling option <opt>
 <&> will bring you back to GENERAL MENU without more changes
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

As there is no need to use these options normally and the menu text is self-explaining, no further description will be given. Note that all options are logical switches and may be enabled and disabled the same way as shown for the last menu. Entering `*` will bring you to the last derivative submenu.

4.4.3 Relax Options

Program `relax` has a huge variety of options to control its actions which in program `define` are grouped together in eight consecutive menus. These are only briefly described in the following sections; for a more detailed discussion of the underlying algorithms refer to the documentation of program `relax` (see Section 5.3). Only experts should try to change default settings.

Optimization Methods

The first of the `relax` subgenus deals with the type of optimization to be performed:

```
------------------------------------------------------------------------
 optimization options for RELAX
------------------------------------------------------------------------
 option | status | description : optimization refers to
------------------------------------------------------------------------
 int    |    F   | INTERNAL coordinates
 crt    |    F   | CARTESIAN coordinates
 bas    |    F   | BASIS SET exponents/scale factors
 glb    |    F   | GLOBAL scaling factor
------------------------------------------------------------------------
 use <opt> for enabling, -<opt> for disabling option <opt>
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

You can choose between a geometry optimization in the space of internal coordinates (in this case you will need definitions of internal coordinates, of course) or in the space of Cartesian coordinates (these possibilities are mutually exclusive, of course). Furthermore optimizations of basis set parameters (exponents, contraction coefficients and scaling factors) or of a global scaling factor is possible (these options are also exclusive, but can be performed simultaneous to a geometry optimization). For the geometry optimization you should normally use internal coordinates as they provide better convergence characteristics in most cases.

Coordinate Updates

The next submenu deals with the way `relax` updates the old coordinates. You may choose a maximum change for the coordinates or you can allow coordinate updates by means of extrapolation:

```
------------------------------------------------------------------------
 coordinate update options for RELAX
------------------------------------------------------------------------
 dqmax <real> : coordinates are allowed to change by at most
                <real> (DEFAULT : 0.3000    ) a.u.
 polish       : perform an interpolation or extrapolation of
                coordinates (DEFAULT :y)
 -polish      : disable inter/extrapolation
------------------------------------------------------------------------
 <RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

These options result in better convergence of your optimization in most cases.

Interconversion Between Internal and Cartesian Coordinates

The interconversion between internal and Cartesian coordinates is not possible directly (in this direction). Instead it is performed iteratively. The following options control this conversion:

```
------------------------------------------------------------------
interconversion options for RELAX
------------------------------------------------------------------
option    | description
------------------------------------------------------------------
on        | switch on interconversion (DEFAULT: off)
qconv <r> | set convergence threshold for interconversion
          | of coordinates to <r>. DEFAULT : <r> =  .1000E-09
iter <i>  | allow at most <i> iterations for interconversion
          | of coordinates. DEFAULT : <i> =   25
crtint    | transform cartesian into internal coordinates (DEFAULT=n)
intcrt    | transform internal into cartesian coordinates (DEFAULT=n)
grdint    | transform cartesian into internal gradients (DEFAULT=n)
hssint    | transform cartesian into internal hessian (DEFAULT=n)
------------------------------------------------------------------
use -<opt> for disabling any interconversion option
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

The options `qconv` and `iter` are used in each normal `relax` run to determine the characteristics of the back-transformation of coordinates into the internal space. With the other options and *interconversion* switched on, you can force `relax` to perform only the specified coordinate transformation and write the transformed coordinates to file `control`. To achieve this, enter `on` to switch to the transformation-only mode, and one of the last four options, e.g. `crtint`, to specify the desired transformation.

Updating the Hessian

`relax` provides a variety of methods to generate an updated Hessian every cycle. This includes the well known methods such as BFGS, DFP, or MS update methods as well as some less common procedures:

```
------------------------------------------------------------
OPTIONS FOR UPDATING THE HESSIAN
------------------------------------------------------------
 option  | status | description
------------------------------------------------------------
none     |    F   | NO UPDATE (STEEPEST DESCENT)
bfgs     |    F   | BROYDEN-FLETCHER-GOLDFARB-SHANNO UPDATE
dfp      |    F   | DAVIDON-FLETCHER-POWELL UPDATE
bfgs-dfp |    F   | COMBINED (BFGS+DFP) UPDATE
ms       |    F   | MURTAGH-SARGENT UPDATE
schlegel |    F   | SCHLEGEL UPDATE
diagup   |    F   | DIAGONAL UPDATE (AHLRICHS/EHRIG)
multidim |    F   | RANK > 2 BFGS-TYPE UPDATE
ahlrichs |    T   | MACRO : AHLRICHS UPDATE (DEFAULT)
------------------------------------------------------------
USE <opt> FOR ENABLING OPTION <opt> AND THUS DISABLING
ALL OTHER OPTIONS.
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

We recommend to use the default method `ahlrichs` which provides excellent convergency in most cases.

General Boundary Conditions for Update

The force constant matrix will only be updated if least `mingeo` cycles exist. The maximum number of cycles used for the update is specified by the parameter `maxgeo`. Normally the default values provided by `define` need not be changed.

```
DEFINE BOUNDARY CONDITIONS FOR UPDATE
------------------------------------------------------------
mingeo <i> | START UPDATE IF THERE ARE AT LEAST <i> CYCLES
           | DEFAULT : min   3
maxgeo <i> | USE LAST <i> CYCLES FOR UPDATE, DEFAULT : max   4
------------------------------------------------------------
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

Special Boundary Conditions for Ahlrichs and Pulay Updates

For the default update method `ahlrichs` some additional control parameters are available which can be defined in this menu:

```
DEFINE BOUNDARY CONDITIONS FOR AHLRICHS OR PULAY UPDATE
------------------------------------------------------------
option    | description
------------------------------------------------------------
modus <i> | DEFINE MODUS FOR GDIIS PROCEDURE : MINIMIZE
          |  <dq|dq> IF  <i> = 0
          |  <g|dq>  IF  <i> = 1
          |  <g|g>   IF  <i> = 2
          |  <dE>    IF  <i> = 3
          | DEFAULT : <i> = 1
fail <r>  | IGNORE GDIIS IF  <g|dq> /| <g|dq> | IS
          | LARGER THAN -<r>. DEFAULT : <r> = 0.1
------------------------------------------------------------
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

For detailed description consult Section 5.3.

```
------------------------------------------------------------
OPTIONS FOR MANIPULATING THE HESSIAN
------------------------------------------------------------
 option      | description
------------------------------------------------------------
diagonal     | RESTRICT UPDATE TO DIAGONAL-ELEMENTS IF
             | METHOD IS BFGS,DFP OR MS. DEFAULT=n
offreset     | DISCARD OFF-DIAGONAL ELEMENTS. DEFAULT=n
offdamp <r>  | DAMP OFF-DIAGONAL ELEMENTS BY 1/(1+<r>) DEFAULT= 1.000
damp <real>  | DAMP UPDATE BY 1/(1+<real>), DEFAULT= .0000E+00
scale <real> | SCALE INPUT HESSIAN BY <real>, DEFAULT= 1.000
```

```
allow <real> | SCALE INPUT HESSIAN BY <real>/|DE| IF |DE|,
             | THE OBSERVED ABSOLUTE CHANGE IN ENERGY, IS
             | OBEYING THE CONDITION |DE| > <real> > 0.
             | DEFAULT : NO SCALING
min <real>   | DO NOT ALLOW EIGENVALUES OF HESSIAN TO DROP
             | BELOW <real>. DEFAULT= .1000E-02
reset <real> | USE <real> AS A RESET VALUE FOR TOO SMALL
             | EIGENVALUES (CP. min). DEFAULT= .1000E-02
max <real>   | DO NOT ALLOW EIGENVALUES OF HESSIAN TO BECOME
             | LARGER THAN <real>. DEFAULT= 1000.
------------------------------------------------------------
WITH THE EXCEPTION OF min,reset AND max, ALL OPTIONS MAY BE
DISABLED BY ENTERING -<opt>
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

Initialization of the Hessian

Finally there are some options to control the choice of the initial Hessian during your geometry optimization:

```
------------------------------------------------------------
FORCE CONSTANTS INITIALIZATION OPTIONS FOR RELAX
------------------------------------------------------------
OPTION    | DESCRIPTION
------------------------------------------------------------
off       | switch off initialization (DEFAULT: on)
cart      | use analytical cartesian hessian provided by a
          | 2nd derivatives calculation. DEFAULT(n)
diag      | use diagonal matrix with diagonal elements set
          | individually within data groups $intdef or
          | $basis or $global. DEFAULT(n)
unit <r>  | use multiple of the unit matrix ( H = <r>*E ).
          | DEFAULT(n) - DEFAULT <r> =  1.000
------------------------------------------------------------
NOTE THAT THESE OPTIONS ARE MUTUALLY EXCLUSIVE
<RETURN> OR * OR q(uit) WILL TERMINATE THIS MENU
```

Option `off` will be used if you have already a good Hessian from a previous calculation which may be used. `cart` describes an even better state where you have a Hessian from a calculation of the second derivatives available (`aoforce`). The other two options describe real procedures for initialization of the Hessian. Default values: stretches (0.5), angles (0.2).

4.4.4 Definition of External Electrostatic Fields

This submenu allows you to calculate first and second numerical derivatives of the energy with respect to an external electric field. The first three options should be clear; `1st` and `2nd` are logical switches which are turned on and off the usual way (`1st` or `-1st`) and `delta` is the increment for the numerical differentiation, that is,

the finite value of the external field, which replaces the (ideally) differential field:

```
------------------------------------------------------------------
electrostatic field definition menu
------------------------------------------------------------------
option       | status | description
------------------------------------------------------------------
1st          |    F   | numerical 1st derivative dE/dField
2nd          |    F   | numerical 2nd derivative d2E/dField2
delta <real> |        | increment for numerical differentiation
             |        | DEFAULT =    .5000E-02
geofield     |    F   | geometry optimization with external field
man          |    F   | explicit definition of electrostatic field(s)
------------------------------------------------------------------
```

`geofield` gives the possibility to perform a whole geometry optimization under the influence of a finite external field and thus to obtain the (distorted) minimum geometry in this field. To do this, an external electrostatic field must be defined explicitly which can be done using command `man`. Note that `geofield` must also be switched on if any properties are to be evaluated in the presence of an electric field. The most prominent example is the calculation of hyperpolarizabilities.

Take Care, due to some inconsistencies in `define` it is *always* necessary to switch on the field calculations manually. Therefore edit the `control` file after having finished your `define` session and enter `on` after the entries of `fields` and `geofield`.

4.4.5 Properties

The program `moloch` used for this purpose is currently being revamped, and will then be much simpler to use. The subsequent description for an older version may not work in all cases—sorry for that.

If you enter `prop` in the general menu, `define` first will check whether the data group `$properties` does already exist in your `control` file or in a file referenced therein. If this is not the case you will be asked to specify the file on which `$properties` shall be written:

```
data group $properties has not yet been specified
FOR INITIALIZING <moloch> KEYWORDS ENTER
  [return] : WRITE TO CONTROL FILE control (DEFAULT),   OR
  filename : WRITE TO ANOTHER FILE
```

Afterwards you will get the following submenu which allows you to control all possible actions of program `moloch`:

```
switch on one or more of the following options <i>
<i> = 1,..., 9
for switching off option <i>, specify -<i>
( 1) trace                                    off
( 2) moments                                  off
( 3) potential                                off
( 4) cowan-griffin                            off
( 5) localization                             off
( 6) population analyses                      off
( 7) plot                                     off
( 8) firstorder                               off
selecting an already active option indicates that
suboptions shall be modified
* or q(uit) = quit | for help, type help <integer>
```

All options in this menu are selected by entering their number as indicated in the first column. For example, to switch on option `trace` enter `1`. The flag `off` will then change to `active`. To switch off an option enter its negative number, e.g. `-1` for `trace`. Most of the options require additional input and will therefore lead you to further submenus. These are briefly described below.

Option `trace`

`trace` will calculate the trace of density times overlap matrix:

$$N = \mathrm{tr}\{\mathbf{DS}\}$$

If the orbitals are orthonormal, N should yield the total number of electrons in your molecule. If this is not true, your MO-vector will most probably be erroneous. For example, the vector might belong to another geometry or basis set. As this is a very sensitive test for errors like these and the calculation requires almost no time, you should always switch on this option.

Option `moments`

This option leads you to the following submenu:

```
add/change options for data group $moments
option            | status | description
------------------|--------|-------------------------------
point <x> <y> <z> |    T   | reference point = (x,y,z)
atom <i>          |    F   | reference point = atom no. <i>
0th               |    T   | compute 0th moment
1st               |    F   | compute 1st moment
2nd               |    F   | compute 2nd moment
3rd               |    F   | compute 3rd moment
------------------|--------|-------------------------------
-<moment>   : skip computation of <moment>
* or q(uit) : terminate input
```

This menu serves to specify the electrostatic moments to be calculated (`0th`=charge, `1st`=dipole moment, `2nd`=quadrupole moment, `3rd`=octuple moment). The reference point is the origin of the coordinate system used in the calculation. The value of any calculated moment will be independent of this reference point, if all lower moments are zero. The default for the reference point is the origin, i.e. the coordinate system used for the calculation of the moments will be the same as the one in which the atomic coordinates are specified. The reference point may be changed by typing `point` with the three new coordinates appended. Alternatively you may choose the coordinates of one of the atoms as reference point by entering `atom` and the atom index.

Option `potential`

This option collects all possible quantities related to the electrostatic field created by the molecular charge distribution. This includes the following suboptions:

```
list of suboptions :
pot          - electrostatic potential
fld          - electrostatic field
fldgrd       - electrostatic field gradient
shld         - diamagnetic shielding
file         - file reference
*            - quit
```

The meaning of the four suboptions `pot`, `fld`, `fldgrd` and `shld` will probably present no problems to you. For each of them, however, you will have to specify at which point(s) this property should be calculated. This is accomplished by one or more data groups `$points` in file `control`. After you chose one or more of the above options, you will therefore reach the next submenu which deals with the specification of these data groups:

```
there are     1 data groups $points
manipulate data group(s) $points
a             - add another data group
m <integer>   - modify <integer>th data group
m all         - modify all data groups
d <integer>   - delete <integer>th data group
d all         - delete all data groups
off <integer> - switch off <integer>th data group
off all       - switch off all data groups
on <integer>  - switch on <integer>th data group
on all        - switch on all data groups
s             - scan through data groups
*             - quit
```

The first line informs you how many of these data groups already exist in your `control` file. Each of these data groups may consist of several points at which the properties will be calculated. You may now create new data groups, delete old

ones or simply switch on or off individual data groups (without deleting them from `control`). The number of different data groups `$points` as well as the number of points in each of them are not limited. However, if you use many points, you should consider specifying them in a separate file. This is most easily done using option `file` in the `potential` menu. This option will create a file for your data groups `$points` and will write a reference of this file to file `control`.

Option `cowan-griffin`

This option activates the computation of the first order relativistic correction to the energy as given by the expectation value of the Cowan–Griffin operator.

Option `localization`

Specifying option `localization` will switch on a Boys localization of molecular orbitals. `define` by default chooses a set of MOs to be localized according to a certain threshold for the orbital energy. Information about these are displayed like this:

```
BOYS localization will be performed with respect to x y z
number of sweeps =      10000
subset of molecular orbitals to be localized :
---> all occupied molecular orbitals
     with orbital energy above -2.00000     Hartree
------------------------------------------------------------------
 shells to be localized
------------------------------------------------------------------
a1       4-5                                          #   1-   5
e        2                                            #   1-   2
------------------------------------------------------------------
you are employing default options for localization
do you want to modify them ? DEFAULT(n)
```

If you want to change the MO selection or other options for the localization enter `y` at this point (By default or when typing `n` you will reach the `moloch` options menu again). You will then be asked whether to change the MO selection method. If you want this, you will enter a little submenu where you can choose one of three possible selection procedures:

`all` selects all occupied orbitals

`thr` selects all occupied orbitals with orbital energy larger than a certain threshold

`man` enables you to select the MOs manually later in this section

If the selection method `thr` is specified you then will be asked for the threshold to be applied for the selection. Afterwards you have the possibility to change some other topics concerning the localization:

- specify other localization directions
- switch on utilization of localized orbitals for population analysis and/or preparation of plot data within the same `moloch` run
- set the maximum number of sweeps in the localization procedure
- specify a file where localized orbitals shall be written to

Option `population analyses`

When activating this option you first have to specify whether the population analysis (PA) should be performed in the CAO (default) or AO basis. Afterwards `define` will ask you whether you want to perform a Mulliken population analysis. In this case, the following submenu will be displayed:

```
add or delete one or more special options for a
mulliken population analysis
 option | status | description
 -------|--------|--------------------------------------
 spdf   |    F   | compute MO contributions to atomic
        |        | brutto populations
 molap  |    F   | compute MO contributions to atomic
        |        | overlap populations
 netto  |    F   | compute atomic netto populations
 irpspd |    F   | compute IRREP contributions to atomic
        |        | brutto populations
 irpmol |    F   | compute IRREP contributions to atomic
        |        | overlap populations
 mommul |    F   | print electrostatic moments resulting
        |        | from atomic charges
 -------|--------|--------------------------------------
 -<option>   : switch off <option>
 * or q(uit) : leave this menu
```

Here you can activate several optional quantities to be computed along with the Mulliken PA. To switch on one or more of these options you must enter the corresponding option keywords, e.g. `spdf netto` for computation of atomic neto populations and MO contributions to atomic brutto populations. The status flags for these tasks will then change from `F` (false) to `T` (true). To switch off any option you simply have to enter the corresponding keyword preceded by a '`-`', e.g. `-netto` for disabling calculation of atomic netto populations.

After having left the Mulliken PA section you will be asked whether a population analysis based on occupation numbers (a modified Roby–Davidson PA) should be performed by `moloch`. When typing `y` you will see the following submenu, where you can switch on several special options for the PA in the same manner as described above.

```
add or delete one or more special options for a
population analysis based on occupation numbers
 option  | status | description
 --------|--------|---------------------------------------
 momao   |    F   | compute MO contributions to modified
         |        | atomic orbital (MAO) occupation numbers
 maodump |    F   | dump all MAOs onto standard output
 maofile |    F   | write MAOs onto a separate file
 select  |    F   | write only those MAOs which have been
         |        | employed in the population analysis
 all     |    F   | write all MAOs
 --------|--------|---------------------------------------
 note that the options select and all are complementary
 -<option>   : switch off <option>
 * or q(uit) : leave this menu
```

Afterwards you have the possibility to change the criterion to be applied for the selection of modified atomic orbitals (MAOs) within the following little submenu:

```
global criterion for selection of Modified Atomic Orbitals (MAOs) :
-------------------------------------------------------------------
 MAOs are employed if 'atomic' density eigenvalues
 exceed a threshold of  .1000
-------------------------------------------------------------------
 specify the appropriate option if you want to use another
 global criterion for selecting MAOs
 option  | status | description
 --------|--------|---------------------------------------
 eig <r> |    T   | select by eigenvalues of the
         |        | 'atomic' density matrices
 occ <r> |    F   | select by occupation numbers
 --------|--------|---------------------------------------
 <r> is the selection threshold (DEFAULT= .1000    )
 * or q(uit) : leave this menu
```

The criterion applied by default is the so-called *atomic density eigenvalue* with a threshold of 0.1. You can switch the criterion to *occupation numbers* by entering `occ`. If you also want to change the threshold, you just have to append its new value to the selection keyword, e.g. `occ .2`. Finally you can select or disable various options in connection with the computation of shared electron numbers (SEN) within the following menu:

```
actual settings for data group $shared electron numbers
 2-center shared electron numbers will be computed;
 values are printed if absolute value exceeds    .0100
 3-center shared electron numbers will be computed;
 values are printed if absolute value exceeds    .0100
 4-center shared electron numbers will be computed;
 values are printed if absolute value exceeds    .0100
add or delete one or more options for the
computation of Shared Electron Numbers (SEN)
 option  | status | description
 --------|--------|---------------------------------------
 2c <r>  |    T   | compute 2-center SEN and print if
         |        | |SEN| > <r> (DEFAULT =  .1000E-01)
 3c <r>  |    T   | compute 3-center SEN and print if
         |        | |SEN| > <r> (DEFAULT =  .1000E-01)
 4c <r>  |    T   | compute 4-center SEN and print if
         |        | |SEN| > <r> (DEFAULT =  .1000E-01)
 --------|--------|---------------------------------------
 nosym   |    F   | switch off use of symmetry
 orbs    |    F   | compute orbital contributions to SEN
 irreps  |    F   | compute irrep contributions to SEN
 --------|--------|---------------------------------------
 -<option>   : switch off <option>
 * or q(uit) : leave this menu
```

The procedure for changing the options is the same as described above. By default calculation of 2-, 3- and 4-center SENs will be enabled with thresholds of 0.01 each.

Option `plot`

This option allows you to prepare the data needed for contour plots of orbital amplitudes or total electron densities. We do not recommend to prepare plotting data this way; an easier method—with an easier syntax—is to generate these data directly by the programs, where densities (also MP2 or excited ones) and Molecular orbitals are calculated. This is described in Chapter 12. If you nevertheless want to prepare the input for plotting data as needed by `moloch` using `define`, on activating `plot` you get the following menu:

```
there are     1 data groups $grid
manipulate data group(s) $grid
 a             - add another data group
 m <integer>   - modify <integer>th data group
 m all         - modify all data groups
 d <integer>   - delete <integer>th data group
 d all         - delete all data groups
 off <integer> - switch off <integer>th data group
 off all       - switch off all data groups
 on <integer>  - switch on <integer>th data group
 on all        - switch on all data groups
 s             - scan through data groups
 *             - quit
```

The commands in this menu serve for the manipulation of data groups `$grid` in an analogous way as described for `$points` in the *potential* section above. `$grid` data groups contain the input information necessary to create the plot data by `moloch` (one data group for each plot). If you want to add a new data group you will enter this submenu:

```
specify the input orbital / input density :
mo <label>    - use occupied molecular orbital <label>
mo density    - use one electron density built from the
                occupied molecular orbitals
lmo <i>       - use localized molecular orbital no. <lmo>
mao <i> <k>   - use modified atomic orbital no. <i>
                centered on atom no. <k>
help          - explanation of the syntax for <label>
*             - quit
```

Here you may specify the orbital to be plotted. To plot the amplitude of the fifth orbital in irrep `a1`, e.g., you would enter `mo 5a1`. Equivalently you can use localized orbitals from a Boys localization procedure or modified atomic orbitals as obtained in a Roby–Davidson–Ahlrichs–Heinzmann population analysis. In the latter cases you will not have to enter an irrep label, as these orbitals are necessarily in C_1 symmetry. Instead you will have to enter the index of the orbital to be plotted (and for option `mao` the index of the atom at which it is situated). In all cases you will additionally have to specify the plane in which the amplitudes or densities will be monitored. To do this, you have to declare two vectors which span that plane and the origin of this new coordinate system relative to the one in which the atomic coordinates are given. Furthermore, you will have to create a grid of points on this plane. The orbital amplitude or electron density will then be calculated for every point in this grid. The grid is created by telling `define` the range to be included along both vectors spanning the plane (where the unit in each direction is the length of the corresponding basis vector) and the number of points to be calculated in this range. It is advantageous to use a wide grid while you test the ranges or planes which give the best results and then to switch to a finer grid for the final calculation. Finally input (MO vector) and output (plot data) files can be specified.

In case you do not want to add a new data group as described above but to change

an existing one, you will be asked which one of the specifications you want to modify.

Chapter 5

Calculation of Molecular Structure and *Ab Initio* Molecular Dynamics

5.1 Structure Optimizations using the Jobex Script

In its normal mode of operation, the shell script `jobex` controls and executes automatic optimizations of molecular geometry parameters. It will cycle through the direct SCF, gradient and force relaxation programs and stop if either the maximum number of cycles is reached or the convergence criteria (change in the total energy, maximum norm of the gradient) are fulfilled. By default, the executable programs are taken from the load modules library within the `TURBOMOLE` directory.

5.1.1 Options

Given a shell the usage is:

```
nohup jobex &
```

This command invokes structure optimization using the default program `statpt`. Structure optimizations using program `relax` can be performed using `-relax` flag:

```
nohup jobex  -relax &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. `jobex` accepts the following arguments controlling the level of calculation, convergence criteria and many more (for example `nohup jobex -gcart 4 &`):

`-energy` *integer* converge total energy up to $10^{(-<\text{integer}>)}$ Hartree (default: 6)

`-gcart` *integer*	converge maximum norm of cartesian gradient up to $10^{(-<\text{integer}>)}$ atomic units (default: 3)
`-c` *integer*	perform up to *integer* cycles (default: 20)
`-dscf`	begin with a direct SCF step
`-grad`	begin with a gradient step
`-statpt`	begin with a force relaxation step
`-relax`	use the `relax` program for force relaxation
`-trans`	perform transition state search
`-level` *level*	define the optimization level, *level*=`scf`, `mp2`, `cc2`, or `uff` (default is `scf`).
`-ri`	use RI modules `ridft` and `rdgrad` (fast Coulomb approximation) instead of `dscf` and `grad` as well as `rimp2` instead of `mpgrad`
`-rijk`	in connection with '-level cc2', the RI-JK versions of HF and CPHF are switched on
`-ex`	perform excited state geometry optimization using `egrad`
`-l <path>`	employ programs from directory `<path>`
`-ls <path>`	load scripts from directory `<path>`
`-md`	a molecular dynamics (MD) run (using `frog` instead of `relax`)
`-mdfile` *file*	commands for MD run are contained in this file (default: `mdmaster`).
`-mdscript` *file*	option to execute a shell script before the `frog` step
`-keep`	keep program output from all optimization steps
`-help`	shows a short description of the commands above

5.1.2 Output

There will be an output written to file `job.start` which informs you about the current options. The convergence is signalled by the file `converged`; otherwise, you should find the file `not.converged` within your working directory. If `jobex` finds a file named `stop` or `STOP` in the working directory, `jobex` will stop after the present step has terminated. You can create `stop` by the command `touch stop`.

The output of the last complete cycle is written to file `job.last`, while the output of the running cycle is collected within the file `job.<cycle>`, where `<cycle>` is the index of the cycle. The convergence criteria and their current values are written out at the bottom of the `job.last` file.

5.2 Program STATPT

5.2.1 General Information

Stationary points are places on the potential energy surface (PES) with a zero gradient, i.e. zero first derivatives of the energy with respect to atomic coordinates. Two types of stationary points are of special importance to chemists. These are minima (reactants, products, intermediates) and first-order saddle points (transition states).

The two types of stationary points can be characterized by the curvature of the PES at these points. At a minimum the Hessian matrix (second derivatives of energy with respect to atomic coordinates) is positive definite, that is the curvature is positive in all directions. If there is one, and only one, negative curvature, the stationary point is a transition state (TS). Because vibrational frequencies are basically the square roots of the curvatures, a minimum has all real frequencies, and a saddle point has one imaginary vibrational "frequency".

Structure optimizations are most effectively done by so-called quasi-Newton–Raphson methods. They require the exact gradient vector and an approximation to the Hessian matrix. The rate of convergence of the structure optimization depends on anharmonicity of the PES and of the quality of the approximation to the Hessian matrix.

The optimization procedure implemented in `statpt` belongs to the family of quasi-Newton–Raphsod methods [25]. It is based on the restricted second-order method, which employes Hessian shift parameter in order to control the step length and direction. This shift parameter is determined by the requirement that the step size should be equal to the actual value of the trust radius, `tradius`, and ensures that the shifted Hessian has the correct eigenvalue structure, all positive for a minimum search, and one negative eigenvalue for a TS search. For TS optimization there is another way of describing the same algorithm, namely as a minimization on the "image" potential. The latter is known as TRIM (Trust Radius Image Minimization) [26].

For TS optimizations the TRIM method implemented in `statpt` tries to maximize the energy along one of the Hessian eigenvectors, while minimizing it in all other directions. Thus, one "follows" one particular eigenvector, hereafter called the "transition" vector. After computing the Hessian for your guess structure you have to identify which vector to follow. For a good TS guess this is the eigenvector with negative eigenvalue, or imaginary frequency. A good comparison of different TS optimization methods is given in [27].

Structure optimizations using `statpt` are controlled by the keyword `$statpt` to be present in the `control` file. It can be set either manually or by using the `stp` menu of `define`. The type of stationary point optimization depends on the value of `itrvec` specified as an option within `$statpt`. By default `itrvec` is set to 0, which implies a structure minimization. A value `itrvec` > 0 implies a transition state optimization using the eigenvalue-following TRIM algorithm, where the index of the transition vector is specified by `itrvec`. Note, that `statpt` orders eigenvalues

(and eigenvectors) of the Hessian in ascending order, shifting six (or five in the case of linear molecules) zero translation and rotation eigenvalues to the end.

Note: this order differs from that used for vibrational frequencies in the `control` file, where rotational and translational eigenvalues are not shifted.

By default a structure optimization is converged when all of the following criteria are met:

- the energy change between two optimization cycles drops below the value given by `threchange` (default: 10^{-6} a.u.),
- the maximum displacement element drops below the value given by `thrmax\-displ` (default: 10^{-3} a.u.),
- the maximum gradient element drops below the value given by `thrmaxgrad` (default: 10^{-3} a.u.),
- the root mean square of the displacement elements drops below the value given by `thrrmsdispl` (default: $5 \cdot 10^{-4}$ a.u.),
- the root mean square of the gradient elements drops below the value given by `thrrmsgrad` (defaul:t $5 \cdot 10^{-4}$ a.u.).

The default values for the convergence criteria can be changed using the `stp` menu of `define`. The necessary keywords are described in Section 14.2.15 below.

For structure optimization of minima with `statpt` as relaxation program just use:

```
jobex  &
```

TS optimizations are performed by the `jobex` invokation:

```
jobex  -trans &
```

5.2.2 Hessian matrix

The choice of the initial Hessian matrix has a great effect on the convergence of the structure optimization. At present, there are three choices for the Hessian matrix in `statpt`. For minimization, a diagonal matrix or approximate Hessian matrix from a forcefield calculation using `uff`(see Section 5.4) can be used. For transition state optimizations you have to provide either the "exact" Hessian or results from the lowest eigenvalue search (LES, see Section 10). Note also that you can **calculate the Hessian with a smaller basis set and/or at a lower wavefunction level, and use it for higher level structure optimization**. Usually, a Hessian matrix calculated in a minimal basis using RI-DFT is good enough for all methods implemented in `TURBOMOLE`.

`statpt` automatically takes the best choice of the Hessian from the `control` file. For minimizations it first looks for the exact Hessian and then for the UFF Hessian. If none of them is found it takes the scaled unit matrix. For transition state optimization the exact Hessian has a higher priority than the results of LES.

The results of LES can be used to obtain an initial Hessian matrix for transition state optimizations involving large molecules, where calculation of the full Hessian is too expensive. Note, that LES calculations for `statpt`, in addition to the `$les` keyword require the following keywords to be added *manually* in the `control` file:

```
$h0hessian
$nomw
```

The default Hessian update for minimization is `bfgs`, which is likely to remain positive definite. The `powell` update is the default for transition state optimizations, since the Hessian can develop a negative curvature as the search progresses.

5.2.3 Finding Minima

Simply specify the `$statpt` keyword in the `control` file and run `jobex` as explained above. You can very often speedup the optimization by calculating the initial Hessian matrix using `uff`.

5.2.4 Finding transition states

Locating minima on a PES is straightforward. In contrast, transition state optimization requires much more input. The diagonal guess Hessian will almost never work, so you must provide a computed one. The Hessian should be computed at your best guess as to what the TS should be.

The real trick here is to find a good guess for the transition state structure. The closer you are, the better. It is often difficult to guess these structures. One way to obtain a good guess is to built an approximate TS and to perform a constrained minimization by freezing internal coordinates that change most during the reaction. Alternatively, you can generate several structures intermediate to reactants and products, and compute the energy at each point. The maximum energy structure is usually a good guess for the true TS.

After obtaining a reasonable initial guess for the TS structure you have to perform a vibrational analysis (or LES calculation for a large molecule) and to identify the index of the transition vector to follow during the optimization. Ideally, this is a vector with a negative eigenvalue, or "imaginary" frequency. The best way to find the right vector is to use some graphical interface to visualize vibrations. For a reasonable guess structure there should be one vibration that resembles the reaction under study. Remember that `statpt` uses a different ordering of eigenvalues as compared to the `aoforce` output—six (five) zero eigenvalues are shifted to the end.

There is an important thing to remember at this point. Even such sophisticated optimization methods like TRIM will not replace your own chemical intuition about where transition states may be located. If you need to restart your run, do so with the coordinates which have the smallest RMS gradient. Note that the energy does not have necessarily to decrease in a transition state search (as opposed to minimizations). It is sometimes necessary to do restart several times (including a recomputation of the Hessian) before the saddle point can be located.

Assuming you do find the TS, it is always a good idea to recompute the Hessian at this structure. It is fairly common, especially when using symmetry, that at your "TS" there is a second imaginary frequency. This means that you have not found the correct TS. The proper procedure is to distort the structure along the "extra" imaginary normal mode using the tool `screwer` (see Section 1.5). Very often such a distortion requires also lowering the point group symmetry. The distortion must be large enough, otherwise the next run will come back to the invalid structure.

5.3 Program Relax

5.3.1 Purpose

`relax` drives and controls a non-linear optimization procedure to locate the minimum (or a stationary point) of a function $f(x)$. In `TURBOMOLE` f is always the electronic energy, and the coordinates x will be referred to as *general coordinates*. They include

- cartesian atomic coordinates
- internal atomic coordinates
- exponents, contraction coefficients and scaling factors of basis functions
- a global scaling factor (a common scaling factor for all basis set exponents)

The optimization employs an iterative procedure based on gradients ∇f of the current and, if available, previous iterations. Various procedures can be applied: steepest descent, Pulay's DIIS, quasi–Newton, conjugate gradients, as well as combinations of them. `relax` carries out:

- update of general coordinates
- update of approximate hessians if needed
- conversion of coordinates (internal $\longleftrightarrow$ cartesian)

The mode of operation is chosen by the keywords `$optimize` and `$interconversion` and the corresponding options, which will be described in the following sections.

5.3.2 Optimization of General Coordinates

After gradients G^k have been calculated for coordinates q^k in optimization cycle k, new coordinates (or basis set exponents) q^{k+1} can be obtained from the quasi–Newton update:

$$q^{k+1} = q^k - F^k G^k$$

where F^k is the inverse of an approximate force constant matrix H^k. This method would immediately converge to the equilibrium geometry if F^k would be the inverse of the exact force constant matrix and the force field would be quadratic. In real applications usually none of these requirements is fulfilled. Often only a crude approximation to the force constant matrix H^k is known. Sometimes a unit matrix is employed (which means coordinate update along the negative gradient with all coordinates treated on an equal footing).

The optimization of nuclear coordinates in the space of internal coordinates is the default task performed by `relax` and does not need to be enabled. Any other optimization task requires explicit specifications in data group `$optimize`, which takes several possible options:

`$optimize` *options*

`internal on/off`	Structure optimization in internal coordinates.
`redundant on/off`	Structure optimization in redundant coordinates.
`cartesian on/off`	Structure optimization in cartesian coordinates.
`basis on/off`	Optimization of basis set exponents, contraction coefficients, scaling factors.
`global on/off`	Optimization of global scaling factor for all basis set exponents.

Note: All options except `internal` are switched off by default, unless they have been activated explicitly by specifying `on`.

Some of the options may be used simultaneously, e.g.

- `internal`, `basis`
- `internal`, `global`
- `cartesian`, `basis`

Other options have to be used exclusively, e.g.

- `internal`, `cartesian`
- `basis`, `global`

The update of the coordinates may be controlled by special options provided in data group `$coordinateupdate` which takes as options:

`dqmax=`*real*	Maximum total coordinate change (default: 0.3).
`interpolate on/off`	Calculate coordinate update by inter/extrapolation using coordinates and gradients of the last two optimization cycles (default: `interpolate on`) if possible.
`statistics` *integer*`/off`	Display optimization statistics for the *integer* previous optimization cycles. Without *integer* all available information will be displayed. `off` suppresses optimization statistics.

The following data blocks are used by program `relax`:

1. Input data from gradient programs `grad`, `rdgrad`, `egrad`, `rimp2`, `mpgrad`, etc.:

`$grad`	cartesian atomic coordinates and their gradients.
`$egrad`	exponents and scale factors and their gradients.
`$globgrad`	global scale factor and its gradient.

2. Input data from force constant program `aoforce`:

`$grad`	cartesian atomic coordinates and their gradients.
`$globgrad`	global scale factor and its gradient.
`$hessian`	the force constant matrix in the space of cartesian coordinates.

3. Output data from program `relax`:

`$coord`	cartesian atomic coordinates.
`$basis`	exponents and scale factors.
`$global`	global scale factor.

For structure optimizations the use of (redundant) internal coordinates is recommended, see Section 4.0.4. Normally internal coordinates are not used for input or output by the electronic structure programs (`dscf`, `mpgrad`, etc.). Instead the coordinates, gradients, etc. are automatically converted to internal coordinates by `relax` on input and the updated positions of the nuclei are written in cartesians coordinates to the data group `$coord`. Details are explained in the following sections.

5.3.3 Force Constant Update Algorithms

In a Newton-type geometry update procedure often only a crude approximation to the force constant matrix H^k is available. What can be done then is to update $F^k = (H^k)^{-1}$ in each iteration using information about previous coordinates and gradients. This constitutes the quasi–Newton or variable metric methods of which there are a few variants:

1. Murtagh/Sargent (MS):

$$F^k = F^{k-1} + \frac{Z^{k-1}(Z^{k-1})^\dagger}{(Z^{k-1})^\dagger dG^{k-1}}$$

2. Broyden/Fletcher/Goldfarb/Shanno (BFGS):

$$F^k = F^{k-1} + \frac{S(dq^{k-1})^\dagger dq^{k-1} - dq^{k-1}(dG^{k-1})^\dagger F^{k-1} - F^{k-1}dG^{k-1}(dq^{k-1})^\dagger}{S1}$$

3. Davidon/Fletcher/Powell (DFP):

$$F^k = F^{k-1} + \frac{(dq^{k-1})^\dagger dq^{k-1}}{S1} - \frac{F^{k-1}dG^{k-1}(dG^{k-1})^\dagger F^{k-1}}{(S-1)S1}$$

4. combined method (BFGS/DFP): If $S1 < (S-1)S1$ and $S1 > 0$ perform DFP update, otherwise BFGS.

The meaning of the symbols above is as follows:

$F^k = (H^k)^{-1}$	approximate inverse force constant matrix in the k-th iteration.s
q^k	general coordinates in the k-th iteration.
G^k	gradients in the k-th iteration.

$dq^{k-1} = q^k - q^{k-1}$

$dg^{k-1} = g^k - g^{k-1}$

$Z^{k-1} = dq^{k-1} - F^{k-1}dG^{k-1}$

$S1 = (dq^{k-1})^\dagger dg^{k-1}$

$S = 1 + ((dg^{k-1})^\dagger F^{k-1}dG^{k-1})/(S1)$

An alternative is to use update algorithms for the hessian H^k itself:
Ehrig, Ahlrichs : *Diagonal* update for the hessian by means of a least squares fit

$$H_{ii}^k = \sqrt{H_{ii}^{k-1}(h_i + d_i)}$$

with the new estimate h for the diagonal elements obtained by

$$h_i = \frac{\sum_k dG_i^k dq_i^k}{\sum_k (dq_i^k)^2}$$

and the error d obtained by the regression

$$d_i = \frac{\sqrt{\frac{\sum_k (dg_i^k)^2}{\sum_k (dq_i^k)^2} - h_i^2}}{k-2}\,.$$

Another alternative is to use DIIS-like methods: structure optimization by direct inversion in the iterative subspace. (See ref. [28] for the description of the algorithm). The DIIS procedure can often be applied with good success, using static or updated force constant matrices.

Any of the algorithms mentioned above may be chosen. Recommended is the macro option `ahlrichs`, which leads to the following actions (n is the maximum number of structures to be included for the update, default is $n = 4$):

ncycles $< n$: geometry update by inter/extrapolation using the last 2 geometries.

ncycles $\geq n$: diagonal update for the hessian as described above; DIIS–like update for the geometry.

$||G|| <$ thr: BFGS-type update of the hessian and quasi–Newton update of (generalized) coordinates.

References for the algorithms mentioned above: [29, 25, 30, 28, 31, 32]

5.3.4 Definition of Internal Coordinates

If structure optimizations are to be performed in the space of internal coordinates (`$optimize internal`, is the default setting), appropriate internal coordinate definitions have to be provided on data block `$intdef`. The types available and their definitions are described in Section 4.1.2. For recommendations about the choice of internal coordinates consult ref. [19]. Nevertheless the structure of `$intdef` will shortly be described. The syntax is (in free format):

```
 1   k   1.00000000   bend  1  2  3    val=1.9500  fdiag=.6666
```

The first items have been explained in Chapter 4.

Two additional items `val=`*real*, `fdiag=`*real* may be supplied for special purposes:

`val=` serves for the input of values for internal coordinates for the interconversion internal → cartesian coordinates; it will be read in by `relax` if the flag for interconversion of coordinates has been activated (`$interconversion on`), or by the interactive input program `define` within the geometry specification menu.

`fdiag=` serves for the input of (diagonal) force constants for the individual internal coordinates to initialize `$forceapprox`.

5.3.5 Structure Optimizations Using Internal Coordinates

This is the default task of `relax` (`$optimize internal on` does not need to be specified!) You need as input the data groups :

`$grad` cartesian coordinates and gradients as provided and accumulated in subsequent optimization cycles by the programs `grad`, or `rdgrad` etc.

`$intdef` definitions of internal coordinates.

`$redundant` definitions of redundant coordinates.

Output will be the updated coordinates on `$coord` and the updated force constant matrix on `$forceapprox`. If any non-default force constant update option has been chosen, `relax` increments its counting variables `<numgeo>`, `<numpul>` within command keyword `$forceupdate`. If the approximate force constant has been initialized (`$forceinit on`) `relax` switches the initialization flag to `$forceinit off`. Refer also to the general documentation of `TURBOMOLE`. It is recommended to check correctness of your definition of internal coordinates:

1. Calculate their values for your cartesian start coordinates using the `relax` program (see Section 5.3.11) or within a `define` session.
2. Have a look at the eigenvectors of the $\mathbf{BmB}^{\dagger}$-matrix. Set some '`?`' behind keyword `$intdef`, if there are any eigenvalues close to zero ($< 10^{-2}$ is to be considered bad for small molecules, but there is no general rule) check those internal coordinates for consistency which contribute to the corresponding eigenvector(s)!

5.3.6 Structure Optimization in Cartesian Coordinates

For this task you have to specify:

```
$optimize
  cartesian on
  internal  off
```

These lines switch on the non-default optimization in cartesian coordinates and switch off the optimization in internal coordinates (this has to be done explicitly!). As input data groups you need only `$grad` as provided by on of the gradient programs. For the first coordinate update an approximate force constant matrix is needed in data group `$forceapprox`. Output will be the updated coordinates on `$coord`, and the updated force constant matrix on `$forceapprox`.

The coordinates for any single atom can be fixed by placing an 'f' in the third to eighth column of the chemical symbol/flag group. As an example, the following coordinates specify acetone with a fixed carbonyl group:

```
$coord
    2.02693271108611      2.03672551266230      0.00000000000000      c
    1.08247228252865     -0.68857387733323      0.00000000000000      c f
    2.53154870318830     -2.48171472134488      0.00000000000000      o     f
```

```
   -1.78063790034738     -1.04586399389434      0.00000000000000      c
   -2.64348282517094     -0.13141435997713      1.68855816889786      h
   -2.23779643042546     -3.09026673535431      0.00000000000000      h
   -2.64348282517094     -0.13141435997713     -1.68855816889786      h
    1.31008893646566      3.07002878668872      1.68840815751978      h
    1.31008893646566      3.07002878668872     -1.68840815751978      h
    4.12184425921830      2.06288409251899      0.00000000000000      h
$end
```

5.3.7 Optimization of Basis Sets (SCF only)

For this task you have to specify:

```
$optimize
  basis     on
  internal  off
```

This example would perform only a basis set optimization without accompanying geometry optimization. It is possible, of course, to optimize both simultaneously: Just leave out the last line of the example (`internal off`). Input data groups are:

`$egrad` Basis set exponents, contraction coefficients, scaling factors and their respective gradients as provided and accumulated in subsequent optimization cycles by one of the programs `grad` or `mpgrad`, if `$drvopt basis on` has been set.

`$basis` Description of basis sets used, see Section 4.2.

Output will be the updated basis on `$basis`, and the updated force constant matrix on `$forceapprox`.

For an example, see Section 15.5.

5.3.8 Simultaneous Optimization of Basis Set and Structure

The optimization of geometry and basis set may be performed simultaneously and requires the specification of:

```
$optimize
  internal  on   (or: cartesian on)
  basis     on
```

and needs as input data groups `$grad` and `$egrad`. Output will be on `$coord`, `$basis`, also on `$forceapprox` (updated).

5.3.9 Optimization of Structure and a Global Scaling Factor

Optimization of a global scaling factor is usually *not* performed in geometry optimizations. It is a special feature for special applications by even more special users. As reference see [33].

To optimize the structure and a global scaling factor specify:

```
$optimize
  internal  on   (or: cartesian on)
  global    on
```

You need as input data groups `$grad` and `$globgrad`, the latter contains the global scaling factors and their gradients accumulated in all optimization cycles. Output will be on `$coord`, `$global`, also on `$forceapprox` (updated). Note that for optimization of a global scaling factor a larger initial force constant element is recommended (about 10.0).

5.3.10 Conversion from Internal to Cartesian Coordinates

Due to translational and rotational degrees of freedom and the non-linear dependence of internal coordinates upon cartesian coordinates, there is no unique set of cartesian coordinates for a given set of internal coordinates. Therefore an iterative procedure is employed to calculate the next local solution for a given cartesian start coordinates. This task may be performed using the `relax` program, but it is much easier done within a `define` session.

5.3.11 Conversion of Cartesian Coordinates, Gradients and Force Constants to Internals

To perform this tasks, you have to activate the interconversion mode by

```
$interconversion on
   cartesian --> internal   coordinate gradient hessian
```

Note that any combination of the three options showed is allowed! The default value is `coordinate`, the two other have to be switched on explicitly if desired.

You need as input data groups:

`intdef`	Definitions of (redundant) internal coordinates
`coord`	Cartesian coordinates (for option '`coordinate`')
`grad`	Cartesian coordinates and gradients as provided and accumulated in subsequent optimization cycles by the various gradient programs (for `coordinate` and `gradient`)

`hessian` Analytical force constant matrix (as provided by the force constant program `aoforce`) (only if option `hessian` is specified). The data group `$hessian (projected)` may be used alternatively for this purpose.

All output will be written to the screen except for option `hessian` (output to data group `$forceapprox`)

5.3.12 The m-Matrix

The m-matrix serves to fix position and orientation of your molecule during geometry optimizations. It cannot be used to fix internal coordinates! The m-matrix is a diagonal matrix of dimension $3n^2$ (where n is the number of atoms). Normally m will be initialized as a unit matrix by `relax`. As an example consider you want to restrict an atom to the xy-plane. You then set the m(z)–matrix element for this atom to zero. You can use at most six zero m-matrix diagonals (for linear molecules only five)—corresponding to translational and rotational degrees of freedom. Note that the condition of the $\mathbf{BmB}^{\dagger}$-matrix can get worse if positional restrictions are applied to the m-matrix. m-matrix elements violating the molecular point group symmetry will be reset to one. Non-default settings for m-matrix diagonals of selected atoms have to be specified within data group `$m-matrix` as:

```
$m-matrix
     1   0.0  0.0  0.0
    10   1.0  0.0  0.0
    11   1.0  1.0  0.0
```

5.3.13 Initialization of Force Constant Matrices

The most simple initial hessian is a unit matrix. However, better choices are preferable. For structure optimizations using internal coordinates you may use structural information to set up a diagonal force constant matrix with elements chosen in accord to the softness or stiffness of the individual modes. For detailed information refer to ref. [31]. For optimization of basis set parameters less information is available. When neither data block `$forceapprox` is available nor `$forceinit on` is set, the force constant matrix will be initialized as a unit matrix. Specifying the force constant initialization key `$forceinit on diag=...` will lead to:

`diag=`*real* Initialization with *real* as diagonal elements.

`diag=default` Initial force constant diagonals will be assigned the following default values:

internal coordinates	:	stretches	0.50
		angles	0.20
scaling factors	:	s,p	1.50
		d	3.00
exponents	:	uncontracted	0.15
		contracted	10.00
contraction coefficients	:		100.00
global scaling factor	:		15.00
cartesian force constants	:		0.50

`diag=individual` Initial force constant diagonals will be taken from
`$intdef fdiag=...` or
`$global fdiag=...`
Similar initialization modes are NOT supported for geometry optimization in cartesian space and for the optimization of basis set parameters!

`carthess` Data group `$hessian` (projected) is used.

5.3.14 Look at Results

The `energy` file includes the total energy of all cycles of a structure optimization completed so far. To get a display of energies and gradients use the UNIX command `grep cycle gradient` which yields, e. g. H_2O.

```
cycle =      1    SCF energy =    -76.3432480651   |dE/dxyz| =  0.124274
cycle =      2    SCF energy =    -76.3575482860   |dE/dxyz| =  0.082663
cycle =      3    SCF energy =    -76.3626983371   |dE/dxyz| =  0.033998
cycle =      4    SCF energy =    -76.3633251080   |dE/dxyz| =  0.016404
cycle =      5    SCF energy =    -76.3634291559   |dE/dxyz| =  0.010640
cycle =      6    SCF energy =    -76.3634910117   |dE/dxyz| =  0.000730
```

This should be self-evident. To see the current—or, if the optimization is converged, the final—atomic distances use the tool `dist`. Bond angles, torsional angles etc. are obtained with the tools `bend`, `tors`, `outp`, etc. In the file `gradient` are the collected cartesian coordinates and corresponding gradients of all cycles. The values of the general coordinates and corresponding gradients are an output of `relax` written to `job.<cycle>` of `job.last` within `jobex`. To look at this search for '`Optimization statistics`' in `job.last` or `job.<cycle>`.

5.4 Force Field Calculations

5.4.1 Purpose

`uff` preoptimizes a structure and calculates an analytical Hessian which can be used as a start Hessian in a geometry optimization. This will accelerate the convergence

of an optimizations. For optimizations in cartesian space this will be faster by a factor of two for any molecule.

5.4.2 How to Perform a UFF Calculation

You have to generate cartesian coordinates (file `coord`), nothing else. You can start an single-point calculation calculation by typing

```
uff
```

To start an `uff` geometry optimization, one has to change the number of cycles (parameter `maxcycle`) in the block `$uff` in the file `control`. The ouput is the optimized structure (file `coord`), the analytical gradient (file `uffgradient`) and the analytical cartesian hessian (file `uffhessian0-0`). Furthermore the `control` file will be modified:

```
$forceinit on
  carthess
$uffhessian file=uffhesian0-0
```

These commands have the effect to inititialize the force constant matric for a geometry optimization with the hessian one.

In some cases `uff` cannot recognize the connectivity, then one can specify the connectivity in the file `ufftopology`. The program will calculate the bond, angle, torsion, inverison and non-bonded terms (force field terms) based on the connectivity specified in the topology file.

5.4.3 The UFF implementation

The `uff` implementation follows the paper by Rappé [7]. The energy expression in `uff` is as follows:

$$
\begin{aligned}
\mathrm{E}_{UFF} = & \sum^{N_B} \frac{1}{2} \cdot K_{IJ} \cdot (r - r_{IJ})^2 \qquad (5.1)\\
& + \sum^{N_A} \begin{cases} \frac{K_{IJK}}{4} \left(1 - \cos(2\theta)\right) : \text{linear case} \\ \frac{K_{IJK}}{9} \left(1 - \cos(3\theta)\right) : \text{trigonal planar case} \\ \frac{K_{IJK}}{16} \left(1 - \cos(4\theta)\right) : \text{quadratic planar case} \\ \frac{K_{IJK}}{16} \left(1 - \cos(4\theta)\right) : \text{octahedral case} \\ K_{IJK} \cdot \left(C_0^A + C_1^A \cos\theta + C_2^A \cos(2\theta)\right) : \text{general case} \end{cases} \\
& + \sum^{N_T} \frac{1}{2} \cdot V_\phi \cdot \left(1 - \cos\left(n\phi_0\right) \cos(n\phi)\right) \\
& + \sum^{N_I} V_\omega \cdot \left(C_0^I + C_1^I \cos\omega + C_2^I \cos 2\omega\right) \\
& + \sum^{N_{nb}} D_{IJ} \cdot \left(-2 \left(\frac{x_{IJ}}{x}\right)^6 + \left(\frac{x_{IJ}}{x}\right)^{12}\right) \\
& + \sum^{N_{nb}} \frac{q_I \cdot q_J}{\epsilon \cdot x}
\end{aligned}
$$

The Fourier coefficients C_0^A, C_1^A, C_2^A of the general angle terms are evaluated as a function of the *natural* angle θ_0:

$$C_2^A = \frac{1}{4 \sin^2 \theta_0} \qquad (5.2)$$

$$C_1^A = -4 \cdot C_2^A \cos\theta_0 \qquad (5.3)$$

$$C_0^A = C_2^A \left(2 \cos^2 \theta_0 + 1\right) \qquad (5.4)$$

The expressions in the engery term are:

$N_B, N_A, N_T, N_I, N_{nb}$	the numbers of the bond-, angle-, torsion-, inversion- and the non bonded-terms.
K_{IJ}, K_{IJK}	forceconstants of the bond- and angle-terms.
r, r_{IJ}	bond distance and *natural* bond distance of the two atoms I and J.
θ, θ_0	angle and *natural* angle for three atoms $I - J - K$.
C_0^A, C_1^A, C_2^A	Fourier coefficients of the general angle terms.
ϕ, ϕ_0	torsion angle and *natural* torison angle of the atoms $I - J - K - L$.
V_ϕ	height of the torsion barrier.
n	periodicity of the torsion potential.
ω	inversion- or out-of-plane-angle at atom I.

V_ω	height of the inversion barrier.
C_0^I, C_1^I, C_2^I	Fourier coefficients of the inversions terms.
x, x_{IJ}	distance and *natural* distance of two non bonded atoms I and J.
D_{IJ}	depth of the Lennard–Jones potential.
q_I, ϵ	partial charge of atoms I and dielectric constant.

One major difference in this implementation concerns the atom types. The atom types in Rappé's paper have an underscore "_". In the present implementation an sp^3 C atom has the name "C 3" instead of "C_3". Particularly the bond terms are described with the harmonic potential and the non-bonded van der Waals terms with the Lennard–Jones potential. The partial charges needed for electrostatic nonbond terms are calculated with the Charge Equilibration Modell (QEq) from Rappé [34]. There is no cutoff for the non-bonded terms.

The relaxation procedure distinguishes between molecules wih more than 90 atoms and molecules with less atoms. For *small* molecules it consists of a Newton step followed by a linesearch step. For *big* molecules a quasi-Newton relaxation is done. The BFGS update of the force-constant matric is done [35, 36, 29, 37]. Pulay's DIIS procedure is implemented for *big* molecule to accelarate the optimization [38, 28].

The coordinates for any single atom can be fixed by placing an 'f' in the third to eighth column of the chemical symbol/flag group. As an example, the following coordinates specify acetone with a fixed carbonyl group:

```
$coord
    2.02693271108611      2.03672551266230      0.00000000000000      c
    1.08247228252865     -0.68857387733323      0.00000000000000      c f
    2.53154870318830     -2.48171472134488      0.00000000000000      o     f
   -1.78063790034738     -1.04586399389434      0.00000000000000      c
   -2.64348282517094     -0.13141435997713      1.68855816889786      h
   -2.23779643042546     -3.09026673535431      0.00000000000000      h
   -2.64348282517094     -0.13141435997713     -1.68855816889786      h
    1.31008893646566      3.07002878668872      1.68840815751978      h
    1.31008893646566      3.07002878668872     -1.68840815751978      h
    4.12184425921830      2.06288409251899      0.00000000000000      h
$end
```

5.5 Molecular Dynamics Calculations

Ab initio molecular dynamics (MD) can be carried out on the ground state Born–Oppenheimer potential hypersurface. At the start of an MD run the user must specify the initial atomic positions and velocities and give some general instructions

for the run. This is managed by running the interactive program `Mdprep` and generating the command file `mdmaster`. If this is successful, the MD run itself may be started: `jobex -md`. Time is then advanced in steps. The electronic potential energy and its gradients are calculated quantum mechanically at the required coordinates each timestep (as detailed above, e.g. `dscf` and `grad`). The MD program `frog` uses the Leapfrog Verlet algorithm [39] to turn the gradients into new atomic positions and velocities. The atoms thus undergo classical Newtonian dynamics on the *ab initio* potential hypersurface. Trajectory information is recorded in a log file (`mdlog`). It is possible to instruct `frog` to heat or cool the system, use a thermostat for canonical dynamics, conserve total energy or read in new positions or velocities: the appropriate keywords are described in Section 14.2.18 below.

5.6 Counterpoise-Corrections using the Jobbsse Script

The shell script `jobbsse` controls and executes the automatic calculation of the counterpoise correction as it has been formulated by Boys and Bernadi (S. F. Boys and F. Bernardi, Mol. Phys., **19,** 553 (1970)) to estimate the Basis Set Superposition Error (BSSE). For a dimer, the cp-correction takes the form for the monomers A and B:

$$E_{AB}^{CP} = E_{AB} - (E_{A(B)} - E_A) - (E_{B(A)} - E_B)$$

Where parentheses denote ghost basis sets without electrons or nuclear charges. For a timer `jobbsse` used by default the conventional so-called site-site functional counterpoise corrections:

$$E_{ABC}^{CP} = E_{ABC} - (E_{A(BC)} - E_A) - (E_{B(AC)} - E_B) - (E_{C(AB)} - E_C) \quad .$$

`jobbsse` works similar as the `jobex` script: it cycles through the SCF/DFT and, if needed, gradient and force relaxation programs and stops if either the maximum number of cycles is reached or the convergence criteria (change in the total energy, maximum norm of the gradient) are fulfilled. It does either only energy calculations or a full geometry optimization including up to three fragments. By default, the executable programs are taken from the load modules library within the `TURBOMOLE` directory.

Note that you need to set up the fragments (and possibly their symmetries using `define` in the geometry menu beforehand. The general structure of a `jobbsse` calculation is as follows:

1. `bsseenergy` is invoked to generate input files for `define`, which is then used to prepare the control files (including occupation numbers, initial guess MOs, etc.) for the different "ghost" and monomer calculations and shell scripts with commands for calculations on these fragments.

2. `jobbsse` cycles over the supermolecular complex and the fragments and computes the energies and, if requested, gradients for them. Then the counterpoise-corrected results are evaluated and written to the standard data groups ($energy and $grad).

3. For geometry optimizations one of the structure relaxation codes (`statpt` or `relax`) is invoked to update the coordinates and check for convergence. If the structure optimization is not converged `jobbsse` continues with the previous step.

Note, that counterpoise-corrected calculations with `jobbsse` are NOT as black-box as ordinary geometry optimizations with `jobex`. The input generated for the fragments are based on the default occupation numbers obtained from the EHT guess, default assignments for the frozen orbitals, memory, etc. Since this might be different from what is needed (or even fail), it is recommended to let `jobbsse` stop after the initial setup step using the flag `-setup` and to check carefully the assigned basis

sets, occupations number and subsystem symmetries. In particular, for MP2 or CC2 calculations with molecules containing not only the atoms H–Ar also the number of frozen orbitals should be checked, and if neccessary corrected.

5.6.1 Options

Given a shell the usage is:

```
nohup jobbsse &
```

This command invokes cp-correction, and, if needed structure optimization using the default program `statpt`. Note, that the program needs to know which calculation is being done. Structure optimizations using program `relax` can be performed using `-relax` flag:

```
nohup jobbsse -opt -relax &
```

`nohup` means that the command is immune to hangups, logouts, and quits. `&` runs a background command. `jobbsse` accepts the following arguments controlling the level of calculation, convergence criteria and many more (for example `nohup jobex -gcart 4 &`):

`-energy` *integer*	converge total energy up to $10^{(-<\text{integer}>)}$ Hartree (default: 6)
`-gcart` *integer*	converge maximum norm of cartesian gradient up to $10^{(-<\text{integer}>)}$ atomic units (default: 3)
`-c` *integer*	perform up to *integer* cycles (default: 100)
`-gradient`	calculate the gradient as well
`-opt`	optimise the structure
`-relax`	use the `relax` program for force relaxation
`-level` *level*	define the optimization level, *level*=`scf`, `dft`, `mp2`, or `cc2` (default is `scf`). Note that the program needs this input! If the level is DFT, the grid will be automatically set to m4.
`-ri`	use RI modules `ridft` and `rdgrad` (fast Coulomb approximation) instead of `dscf` and `grad` as well as `rimp2` instead of `mpgrad`
`-l <path>`	employ programs from directory `<path>`
`-mem` *integer*	Is able to control the memory from outside `define` Note that if you did not define any memory, it is automatically set to 1 GB

`-trimer`	calculates, in case we have a trimer: Energy = ABC - AB(C) + AB - AC(B) + AC - BC(A) + BC rather than Energy = ABC - A(BC) + A - B(AC) + B - C(AB) + C (note that the first term neglects the BSSE in the dimer)
`-setup`	Interrupt calculation after the initial setup step to check and possibly correct the control files for the fragments and the supermolecule. To continue, start `jobbsse` without the `-setup` option.
`-help`	shows a short description of the commands above

5.6.2 Output

There will be an output written to file `bsse_out`. In this file, you will find all individual energies computed which were used to calculate the last cp-corrected energy. The same holds true for the last gradients, which are written to `grad_out`.

The convergence criteria and their current values are written out at the `not.converged` file. For the possible options to control convergence check the subsection for the optimization program used (`statpt`, which is used by default, or `relax`). Since for weak complexes the force constants for intra- and intermolecular bonds very strongly in magnitude, it is recommended to use whenever possible redundant internal coordinates.

Chapter 6

Hartree–Fock and DFT Calculations

Energy and gradient calculations at the Hartree–Fock (HF) and DFT level can be carried out in two ways: `dscf` and `grad` perform conventional calculations based on four–center two–electron repulsion integrals (ERI's); `ridft` and `rdgrad` employ the RI–J approximation, as detailed below.

`dscf` and `grad` are modules for energy and gradient calculations at the HF and DFT level, which use an efficient semi–direct SCF algorithm. Calculation of the Coulomb and HF exchange terms is based on the conventional method employing four–center two–electron repulsion integrals (ERI's). These modules should be used for HF and DFT calculations with exchange-correlation functionals including HF exchange contribution, e.g. B3–LYP, if further approximations (RI–J) are to be avoided. All functionalities are implemented for closed–shell RHF and open–shell UHF reference wavefunctions. Restricted open shell treatments (ROHF) are supported on the HF level only, i. e. not for DFT.

The most important special features of the `dscf` and `grad` modules are:

- Selective storage of the most time consuming and frequently used integrals. The integral storage is controlled by two threshold parameters, `$thize` and `$thime`, related to integral size and computational cost.
- Efficient convergence acceleration techniques for energy calculations. They include standard methods for convergence acceleration (DIIS), which reduce the number of SCF iterations needed as well as methods to reduce the effort within each iteration when the calculation is almost converged (integral prescreening and differential density scheme).

`ridft` and `rdgrad` are modules for very efficient calculation of energy and gradient at the Hartree–Fock (HF) and DFT level [40]. Both programs employ the Resolution of the Identity approach for computing the electronic Coulomb interaction (RI–J). This approach expands the molecular electron density in a set of atom–centered auxiliary functions, leading to expressions involving three–center ERI's only. This

usually leads to a more than tenfold speedup for non–hybrid DFT compared to the conventional method based on four–center ERI's (for example the `dscf` or `grad` module).

The combination of RI–J for Coulomb–interactions with a case–adapted conventional exchange treatment reduces the scaling behaviour of the (conventional) exchange evaluation required in HF–SCF and hybrid DFT treatments. Usage of `ridft` and `rdgrad` for HF and hybrid DFT is of advantage (as compared to `dscf` and `grad`) for larger systems, where it reduces computational costs significantly.

The most important special features of the `ridft` and `rdgrad` modules are:

- A very efficient semi-core algorithm for energy calculation. The most expensive three–center integrals are kept in memory which significantly reduces the computational time for small and middle sized molecules. The amount of stored integrals is controlled by simply specifying the amount of free memory using the keyword `$ricore`.
- Multipole accelerated RI for Coulomb (MARI–J) linear scaling ($O(N)$) method for large molecules. It significantly reduces calculation times for molecules with more than 1000 basis functions.

All algorithms implemented in `dscf`, `grad`, `ridft`, and `rdgrad` modules can exploit molecular symmetry for *all* finite point groups. Typically, the CPU time is proportional to $1/N_G$, where N_G is the order of the nuclear exchange group. Another important feature is a parallel implementation using the MPI interface.

Additionally `dscf` and `ridft` modules include the following common features:

- An UHF implementation [41] with automatic generation of optimal start vectors by solving the HF instability equations [42] in the AO basis (see the keyword `$scfinstab` for detailed information).
- Occupation number optimization using (pseudo-Fermi) thermal smearing.

RI-techniques can also be used for the Hartree–Fock exchange part of the Fock matrix (RI-HF). This is done by the ridft-module, if the keyword `$rik` is found in the `control` file. In this case ridft performs a Hartree–Fock-SCF calculation using the RI- approximation for both J and K, if suitable auxiliary basis sets (which differ from that used for fitting of the Coulomb part only) are specified. This is efficient only for comparably large basis sets like TZVPP, cc-pVTZ and larger.

Prerequisites

Both `dscf` and `ridft` require the `control` file and starting orbitals obtained from the extended Hückel guess using `define`.

Energy calculations using `dscf` can be performed in a direct or semi-direct mode. In the direct mode all four-center ERI's are recalculated at each SCF iteration. The

semi-direct mode uses a selective storage of the most time consuming and frequently used integrals. The amount of integrals stored is controlled by the keywords `$thize` and `$thime`, related to integral size and computational cost. The semi-direct mode requires a separate `dscf` statistics run to estimate the disk space needed for integral storage. The statistics run requires the keyword `$statistics dscf` to be present in the `control` file. It can be set either manually or using the tool `Stati`.

For `ridft` and `rdgrad` following additional prerequisites are required:

1. An auxiliary basis defined in the data group `$jbas`. This group is created automatically when using `ri` menu of `define`.
2. The maximum core memory the program is allowed to allocate should be defined in the data group `$ricore`; the recommended value is 75–85% of the available (physical) core memory.
3. Calculations using MARI-J method require the keyword `$marij`.
4. For RI-HF-calculations auxiliary bases defined in the data group `$jkbas` are needed. This group is created by the `rijk` menu in `define`.

How to Perform a Calculation

Single point calculations
: Call the `dscf` or `ridft` program after running `define`.

Geometry optimizations and molecular dynamics
: For HF or DFT calculations using `dscf` and `grad` simply invoke `jobex`. For DFT calculations using `ridft` and `rdgrad` type `jobex -ri`; see Section 5.1 for additional options and parameters for geometry optimizations and *ab initio* molecular dynamics calculations.

6.1 Background Theory

In Hartree–Fock theory, the energy has the form,

$$E_{HF} = h + J - K + V_{nuc}, \tag{6.1}$$

where h is the one-electron (kinetic plus potential) energy, J is the classical Coulomb repulsion of the electrons, K is the exchange energy resulting from the quantum (fermion) nature of electrons, and V_{nuc} is the nuclear repulsion energy.

In density functional theory, the exact Hartree–Fock exchange for a single determinant is replaced by a more general expression, the exchange-correlation functional, which can include terms accounting for both exchange energy and the electron correlation which is omitted from Hartree–Fock theory. The DFT energy is expressed as a functional of the molecular electron density $\rho(\mathbf{r})$,

$$E_{DFT}[\rho] = T[\rho] + V_{ne}[\rho] + J[\rho] + E_x[\rho] + E_c[\rho] + V_{nuc}, \tag{6.2}$$

where $T[\rho]$ is the kinetic energy, $V_{ne}[\rho]$ is the nuclei-electron interaction, $E_x[\rho]$ and $E_c[\rho]$ are the exchange and correlation energy functionals.

The exchange and correlation functionals normally used in DFT are integrals of some function of the density and possibly the density gradient. In addition to pure DFT methods, `dscf` and `grad` modules support hybrid functionals in which the exchange functional includes the Hartree–Fock exchange, e.g. B3-LYP.

6.2 Exchange-Correlation Functionals Available

The following exchange-correlation functionals are available:

- LDAs: S-VWN, PWLDA
- GGAs: B-VWN, B-LYP, B-P, PBE
- MGGA: TPSS
- hybrid functionals: BH-LYP, B3-LYP, PBE0, TPSSh
- double–hybrid functional: B2-PLYP (energy calculations only!)

In detail, the functional library consists of:

- The Slater–Dirac exchange functional only (S) [43, 44].
- The 1980 correlation functional (functional V in the paper) of Vosko, Wilk, and Nusair only (VWN) [45].
- A combination of the Slater–Dirac exchange and Vosko, Wilk, and Nusair 1980 (functional V) correlation functionals (S-VWN) [43, 44, 45].
- The S-VWN functional with VWN functional III in the paper. This is the same functional form as available in the Gaussian program [43, 44, 45].
- A combination of the Slater–Dirac exchange and Perdew-Wang (1992) correlation functionals [43, 44, 46].
- A combination of the Slater–Dirac exchange and Becke's 1988 exchange functionals (B88) [43, 44, 47].
- Lee, Yang, and Parr's correlation functional (LYP) [48].
- The B-LYP exchange-correlation functional (B88 exchange and LYP correlation functionals) [43, 44, 47, 48].
- The B-VWN exchange-correlation functional (B88 exchange and VWN (V) correlation functionals) [43, 44, 47, 45].

- The B-P86 exchange-correlation functional (B88 exchange, VWN(V) and Perdew's 1986 correlation functionals) [43, 44, 47, 45, 49].
- The Perdew, Burke, and Ernzerhof (PBE) exchange-correlation functional [43, 44, 46, 50].
- The Tao, Perdew, Staroverov, and Scuseria functional (Slater–Dirac, TPSS exchange and Perdew-Wang (1992) and TPSS correlation functionals) [43, 44, 46, 51].

Additionally, for all four modules (`dscf`, `grad`, `ridft`, and `rdgrad`) following hybrid functionals are available (a mixture of Hartree–Fock exchange with DFT exchange-correlation functionals):

- The BH-LYP exchange-correlation functional (Becke's half-and-half exchange in a combination with the LYP correlation functional) [43, 44, 47, 48, 52].
- The B3-LYP exchange-correlation functional (Becke's three-parameter functional) with the form,

$$0.8S + 0.72B88 + 0.2HF + 0.19VWN(V) + 0.81LYP \tag{6.3}$$

  where HF denotes the Hartree-Fock exchange [43, 44, 47, 48, 53].
- The B3-LYP exchange-correlation functional with VWN functional V in the paper. This is the same functional form as available in the Gaussian program.
- The 1996 hybrid functional of Perdew, Burke, and Ernzerhof, with the form,

$$0.75(S + PBE(X)) + 0.25HF + PW + PBE(C) \tag{6.4}$$

  where PBE(X) and PBE(C) are the Perdew–Burke–Ernzerhof exchange and correlation functionals and PW is the Perdew–Wang correlation functional [43, 44, 46, 50, 54].
- The TPSSH exchange-correlation functional of Staroverov, Scuseria, Tao and Perdew with the form,

$$0.(S + TPSS(X)) + 0.1HF + PW + TPSS(C) \tag{6.5}$$

  where HF denotes the Hartree–Fock exchange [43, 44, 46, 51, 55].
- The localized Hartree–Fock method (LHF) to obtain an effective exact exchange Kohn–Sham potential [56, 57] (module `dscf`only).

Additionally the Double-Hydbrid Functional B2-PLYP can be used for single point energy calculations. Note that one has to run an MP2 calculation after the DFT step to get the correct B2-PLYP energy!

B2-PLYP is a so-called double-hybrid density functional (DHDF) [58] that uses in addition to a non-local exchange contribution (as in conventional hybrid-GGAs) also a non-local perturbation correction for the correlation part. Note the following options/restrictions in the present version of this method:

- single point calculations only (computed with the DSCF and RIMP2/RICC2 modules).
- UKS treatment for open-shell cases.
- can be combined with *resolution-of-identity* approximation for the SCF step (RI-JK option).
- can be combined with the dispersion correction (DFT-D method, s_6(B2-PLYP)=0.55).

The non-local perturbation correction to the correlation contribution is given by second-order perturbation theory. The idea is rooted in the *ab initio* Kohn-Sham perturbation theory (KS-PT2) by Görling and Levy [59,60]. The mixing is described by two empirical parameters a_x and a_c in the following manner:

$$\begin{aligned} E_{XC}(DHDF) = (1-a_x)E_X(GGA) + a_x E_X(HF) \\ +(1-a_c)E_C(GGA) + a_c E_C(KS-PT2), \end{aligned} \tag{6.6}$$

where $E_X(GGA)$ is the energy of a conventional exchange functional and $E_C(GGA)$ is the energy of a correlation functional. $E_X(HF)$ is the Hartree-Fock exchange of the occupied Kohn-Sham orbitals and $E_C(KS-PT2)$ is a Møller-Plesset like perturbation correction term based on the KS orbitals:

$$E_C(KS-PT2) = \frac{1}{2}\sum_{ia}\sum_{jb}\frac{(ia|jb)[(ia|jb)-(ib|ja)]}{e_i+e_j-e_a-e_b}. \tag{6.7}$$

The method is self-consistent only with respect to the first three terms in Eq. 6.6, i.e., first a SCF using a conventional hybrid-GGA is performed first. Based on these orbitals $E_C(KS-PT2)$ is evaluated afterwards and added to the total energy.

For B2-PLYP, B88 exchange [47] and LYP correlation [48] are used with the parameters $a_x = 0.53$ and $a_c = 0.27$. Due to the relatively large Fock-exchange fraction, self-interaction error related problems are alleviated in B2-PLYP while unwanted side effects of this (reduced account of static correlation) are damped or eliminated by the PT2 term.

How to use B2-PLYP:

- during preparation of your input with DEFINE select `b2-plyp` in the DFT menu.
- carry out a DSCF run. Prepare and run a RI-MP2 calculation with either RIMP2 or RICC2 program modules.
- the RI-MP2 program directly prints the B2PLYP energy if this functional has been chosen before
- if you use the RICC2 program the scaled ($a_c = 0.27$) second-order correlation energy. must be added manually to the SCF-energy.
- in order to maintain consistency of the PT2 and GGA correlation parts, it is recommend not to apply the frozen-core approximation in the PT2 treatment.

6.3 Restricted Open-Shell Hartree–Fock

6.3.1 Brief Description

The spin-restricted open-shell Hartree–Fock method (ROHF) can always be chosen to systems where all unpaired spins are parallel. The `TURBOMOLE` keywords for such a case (one open shell, triplet e_g^2) are:

```
$open shells  type=1
  eg     1                (1)
$roothaan   1
  a=1  b=2
```

It can also treat more complicated open-shell cases, as indicated in the tables below. In particular, it is possible to calculate the $[xy]^{\text{singlet}}$ case. As a guide for expert users, complete ROHF `TURBOMOLE` input for O_2 for various CSFs (configuration state function) is given in Section 15.6. Further examples are collected below.

The ROHF ansatz for the energy expectation value has a term for interactions of closed-shells with closed-shells (indices k, l), a term for purely open-shell interactions (indices m, n) and a coupling term (k, m):

$$\begin{aligned} E =& 2\sum_k h_{kk} + \sum_{k,l}(2J_{kl} - K_{kl}) \\ &+ f[2\sum_m h_{mm} + f\sum_{m,n}(2aJ_{mn} - bK_{mn}) + 2\sum_{k,m}(2J_{km} - K_{km})] \end{aligned}$$

where f is the (fractional) occupation number of the open-shell part ($0 < f < 1$), and a and b are the Roothaan parameters, numerical constants which depend on the particular configuration of interest.

6.3.2 One Open Shell

Given are term symbols (up to indices depending on actual case and group) and a and b coefficients. n is the number of electrons in an *irrep* with degeneracy n_{ir}. Note that not all cases are Roothaan cases.

All *single electron* cases are described by:

$$a = b = 0$$

Table 6.1: Roothaan-coefficients a and b for cases with degenerate orbitals.

n_{ir}=2: e (div. groups), π, δ ($C_{\infty v}$, $D_{\infty h}$)						
n	f	e^n	π^n	δ^n	a	b
2	1/2	^{3}A	$^3\Sigma$	$^3\Sigma$	1	2
		^{1}E*	$^1\Delta$	$^1\Gamma$	1/2	0
		^{1}A	$^1\Sigma$	$^1\Sigma$	0	-2
3	3/4	^{2}E	$^2\Pi$	$^2\Delta$	8/9	8/9
1 n_{ir}=3: p (O(3)), t (T, O, I)†						
n	f	p^n			a	b
2	1/3	^{3}P			3/4	3/2
		^{1}D**			9/20	$-3/10$
		^{1}S			0	-3
3	1/2	^{4}S			1	2
		^{2}D**			4/5	4/5
		^{2}P			2/3	0
4	2/3	^{3}P			15/16	9/8
		^{1}D**			69/80	27/40
		^{1}S			3/4	0
5	5/6	^{2}P			24/25	24/25
only irrep g(I) (mainly high spin available)						
n	f	g^n			a	b
1	1/8	^{2}G			0	0
2	1/4	††			2/3	4/3
		^{1}A			0	-4
3	3/8	^{4}G			8/9	16/9
4	1/2	^{5}A			1	2
5	5/8	^{4}G			24/25	32/25
6	3/4	††			26/27	28/27
		^{1}A			8/9	4/9
7	7/8	^{2}G			48/49	48/49

continues on next page

Table 6.1: Roothaan-coefficients a and b for cases with degenerate orbitals (continued).

d(O3), h(I) (mainly high-spin cases work)				
n	f	d^n	a	b
1	1/10	^{2}D	0	0
2	1/5	^{3}F+^{3}P††	5/8	5/4
		^{1}S	0	−5
3	3/10	^{4}F+^{4}P††	5/6	5/3
4	2/5	^{5}D, ^{5}H	15/16	15/8
5	1/2	^{6}S, ^{6}A	1	2
6	3/5	^{5}D, ^{5}H	35/36	25/18
7	7/10	^{4}F+^{4}P††	95/98	55/49
8	4/5	^{3}F+^{3}P††	125/128	65/64
		^{1}S	15/16	5/8
9	9/10	^{2}D, ^{2}H	80/81	80/81

* except cases (e.g. D_{2d} or D_{4h}) where e^2 gives only one-dimensional irreps, which are not Roothaan cases.

† only p^n given, the state for groups T_d etc. follows from S → A (T,O,I) P → T (T,O,I) D → H (I), E+T (T,O)

** This is not a CSF in T or O, (a, b) describes average of states resulting from E+T

†† (a, b) describes weighted average of high spin states, not a CSF.

Example

The $4d^9\,5s^2\ ^2$D state of Ag, in symmetry I

```
$closed shells
  a       1-5                                    ( 2 )
  t1      1-3                                    ( 2 )
  h       1                                      ( 2 )
$open shells type=1
  h       2                                      ( 9/5 )
$roothaan         1
  a = 80/81      b = 80/81
```

6.3.3 More Than One Open Shell

A Half-filled shell and all spins parallel

All open shells are collected in a single open shell and

$$a = 1 \qquad b = 2$$

Example: The $4d^5\,5s^1\ {}^7$S state of Mo, treated in symmetry I

```
$roothaan         1
  a = 1      b = 2
$closed shells
  a      1-4                                    ( 2 )
  t1     1-3                                    ( 2 )
  h      1                                      ( 2 )
$open shells type=1
  a      5                                      ( 1 )
  h      2                                      ( 1 )
```

Two-electron singlet coupling

The two MOs **must** have different symmetries (not required for triplet coupling, see example 6.3.3). We have now two open shells and must specify three sets of (a, b), i.e. one for each pair of shells, following the keyword `$rohf`.

Example: CH_2 in the $^1B_2$ state from $(3a_1)^1\ (1b_2)^1$, molecule in (x,z) plane.

```
$closed shells
  a1      1-2                                    ( 2 )
  b1      1                                      ( 2 )
$open shells type=1
  a1      3                                      ( 1 )
  b2      1                                      ( 1 )
$roothaan         1
$rohf
  3a1-3a1  a = 0      b = 0
  1b2-1b2  a = 0      b = 0
  3a1-1b2  a = 1      b = -2
```

Two open shells

This becomes tricky in general and we give only the most important case:

shell 1 is a Roothaan case, see 6.3.2

shell 2 is one electron in an a (s) MO ($n_{ir} = 1$)
with parallel spin coupling of shells.

This covers e.g. the $p^5\,s^1$ ^{3}P states, or the d^4s^1 ^{6}D states of atoms. The coupling information is given following the keyword `$rohf`. The (a, b) within a shell are taken from above (6.3.2), the cross term (shell 1)–(shell 2) is in this case:

$$a = 1 \quad \text{always}$$

$$b = 2 \quad \text{if } n \le n_{ir} \qquad b = \frac{(2n_{ir})}{n} \quad \text{if } n > n_{ir}$$

where n_{ir} and n refer to shell 1.

Example 1: The $4d^4\,5s^1$ ^{6}D state of Nb, in symmetry I

```
$closed shells
 a       1-4                                    ( 2 )
 t1      1-3                                    ( 2 )
 h       1                                      ( 2 )
$open shells type=1
 a       5                                      ( 1 )
 h       2                                      ( 4/5 )
$roothaan         1
$rohf
5a-5a      a = 0      b = 0
5a-2h      a = 1      b = 2
2h-2h      a = 15/16  b = 15/8
```

Example 2: The $4d^5\,5s^1$ ^{7}S state of Mo, symmetry I (see Section 6.3.3) can also be done as follows.

```
$roothaan         1
$rohf
5a-5a      a = 0      b = 0
5a-2h      a = 1      b = 2
2h-2h      a = 1      b = 2
```

```
$closed shells
 a      1-4                                    ( 2 )
 t1     1-3                                    ( 2 )
 h      1                                      ( 2 )
$open shells type=1
 a      5                                      ( 1 )
 h      2                                      ( 1 )
```

The shells $5s$ and $4d$ have now been made *inequivalent*. Result is identical to 6.3.3 which is also more efficient.

Example 3: The $4d^9\,5s^1\ {}^3$D state of Ni, symmetry I

```
$closed shells
 a       1-3                                    ( 2 )
 t1      1-2                                    ( 2 )
$open shells type=1
 a       4                                      ( 1 )
 h       1                                      ( 9/5 )
$roothaan         1
$rohf
 4a-4a a = 0     b = 0
 1h-1h a = 80/81 b = 80/81
 4a-1h a =1      b = 10/9
```

(see basis set catalogue, basis `SV.3D` requires this input and gives the energy you must get)

6.3.4 Miscellaneous

Valence states

Valence states are defined as the weighted average of *all* CSFs arising from an electronic configuration (occupation): $(\mathrm{MO})^n$. This is identical to the average energy of all Slater determinants.

$$a = b = \frac{2n_{ir}(n-1)}{(2n_{ir}-1)n}$$

This covers, e.g. the cases $n = 1$ and $n = 2n_{ir} - 1$: p^1, p^5, d^1, d^9, etc, since there is only a single CSF which is identical to the average of configurations.

Totally symmetric singlets for 2 or ($2n_{ir}$-2) electrons

$$
\begin{aligned}
n &= 2 & a &= 0 \qquad b = -n_{ir} \\
n &= (2n_{ir} - 2) & a &= \frac{n_{ir}(n_{ir}-2)}{(n_{ir}-1)^2} \\
& & b &= \frac{n_{ir}(n_{ir}-3)}{(n_{ir}-1)^2}
\end{aligned}
$$

This covers the ^{1}S states of p^2, p^4, d^2, d^8, etc.

Average of high-spin states: n electrons in MO with degenerate n_{ir}.

$$
\begin{aligned}
a &= \frac{n_{ir}\big(4k(k+l-1)+l(l-1)\big)}{(n_{ir}-1)n^2} \\
b &= \frac{2n_{ir}\big(2k(k+l-1)+l(l-1)\big)}{(n_{ir}-1)n^2}
\end{aligned}
$$

where: $k = \max(0, n - n_{ir})$, $l = n - 2k = 2S$ (spin)

This covers most of the cases given above. A CSF results only if $n = \{1, (n_{ir}-1)$, n_{ir}, $(n_{ir}+1)$, $(2n_{ir}-1)\}$ since there is a single high-spin CSF in these cases.

The last equations for a and b can be rewritten in many ways, the probably most concise form is

$$
\begin{aligned}
a &= \frac{n(n-2)+2S}{(n-2f)n} \\
b &= \frac{n(n-2)+(2S)^2}{(n-2f)n}\,.
\end{aligned}
$$

This applies to shells with one electron, one hole, the high-spin couplings of half-filled shells and those with one electron more ore less. For d^2, d^3, d^7, and d^8 it represents the (weighted) average of high-spin cases: ^{3}F + ^{3}P for d^2,d^8, ^{4}F + ^{4}P for d^3, d^7.

6.4 Two-component Hartree–Fock and DFT Calculations

6.4.1 Background Theory

Two-component treatments allow for self-consistent calculations including spin-orbit interactions. These may be particularly important for compounds containing heavy elements (additionally to scalar relativistic effects). Two-component treatments were implemented within the module `ridft` for RI-JK-Hartree-Fock and RI-DFT (local, gradient-corrected and hybrid functionals) via effective core potentials describing both scalar and spin-orbit relativistic effects. The theoretical background and the implementation is described in [61]. Two-component treatments require the use of complex two-component orbitals

$$\psi_i(\mathbf{x}) = \begin{pmatrix} \psi_i^{\alpha}(\mathbf{r}) \\ \psi_i^{\beta}(\mathbf{r}) \end{pmatrix}$$

instead of real (non-complex) one-component orbitals needed for non-relativistic or scalar-relativistic treatments. The Hartree-Fock and Kohn-Sham equations are now spinor equations with a complex Fock operator

$$\begin{pmatrix} \hat{F}^{\alpha\alpha} & \hat{F}^{\alpha\beta} \\ \hat{F}^{\beta\alpha} & \hat{F}^{\beta\beta} \end{pmatrix} \begin{pmatrix} \psi_i^{\alpha}(\mathbf{r}) \\ \psi_i^{\beta}(\mathbf{r}) \end{pmatrix} = \epsilon_i \begin{pmatrix} \psi_i^{\alpha}(\mathbf{r}) \\ \psi_i^{\beta}(\mathbf{r}) \end{pmatrix}.$$

The wavefunction is no longer eigenfunction of the spin operator, the spin vector is no longer an observable.

In case of DFT for open-shell systems rotational invariance of the exchange-correlation energy was ensured by the non-collinear approach. In this approach the exchange-correlation energy is a functional of the particle density and the absolute value of the spin-vector density $\vec{m}(\mathbf{r})$ ($\vec{\boldsymbol{\sigma}}$ are the Pauli matrices)

$$\vec{m}(\mathbf{r}) = \sum_i \psi_i^{\dagger}(\mathbf{x}) \vec{\boldsymbol{\sigma}} \psi_i(\mathbf{x}).$$

This quantity replaces the spin-density (difference between density of alpha and beta electrons) of non- or scalar-relativistic treatments.

For closed-shell species the Kramers-restricted scheme, a generalization of the RHF-scheme of one component treatments, is applicable.

6.4.2 How to use

The keyword `$soghf` enforces the two-component calculations. Keywords for specification of the method of calculation are the same as for the one-component case (`$dft` and `$rij` for pure density functional calculations within the RI-J-approximation, `$rij` and `$rik` for Hartree-Fock with the RI-approximation for Coulomb and exchange operators, and all three keywords for Hybrid-DFT). The DIIS scheme for

complex Fock operators can be activated by inserting `$gdiis` in the control-file. For closed-shell species a Kramers invariant density functional formalism (only pure density functionals) can be switched on with the keyword `$kramers`. These keywords have to be inserted into the control-file manually.

As start wavefunctions Hückel-, UHF- or RHF-wavefunctions may be used. The two-component formalism does not support the point group symmetries, start wave functions may be transformed to C1 symmetry by `define` or the script 'uhfuse'.

For spin-orbit treatments two-component ECPs (suffix $-2c$) are required, the use of extended basis sets accounting for the spatial splitting of inner p-shells (also suffix $-2c$) is recommended (see [62]). ECPs and basis sets def2-XVP-2c (X=S ,TZ ,QZ) are available for Ag - I, Au - At, they can be selected within the define session. RI-J and RI-JK auxiliary basis sets of def2-type are of sufficient flexibility for two-component treatments; they are the same with and without suffix $-2c$.

The two-component formalism may be most easily prepared and applied in the following way:

- run `define`: choose C1-symmetry; select ECPs and basis sets with suffices $-2c$ for the respective elements. The corrresponding auxiliary basis sets are provided automatically.
- insert `$soghf` in the control file (as well as further desired keywords).
- For open-shell molecules it is often helpful to increase the value for $scforbitalshift closedshell; a value of ca. 1.0 may serve as a rough recommendation.
- start the two-component calculation with `ridft`
- At the end of the SCF procedure real and imaginary parts of spinors are written to files spinor.r and spinor.i, eigenvalues and spinor occupations are collected in the file EIGS, the total energy is added to data group `$energy`. The data groups `$closed shells` (`$alpha shells` and `$beta shells` for open shell cases) are no longer significant, but nevertheless kept in the control-file; additionally the spinor occupations are diposited in data group `$spinor`.

6.5 Using the Douglas–Kroll–Hess (DKH) Hamiltonian

For consideration of scalar-relativistic effects in all-electron calculations the scalar-relativistic Douglas–Kroll–Hess (DKH) Hamiltonian can be employed in the modules `dscf` and `ridft`. Please make sure to use an all-electron basis set (in particular, ECPs are not allowed).

This Hamiltonian is defined up to a certain order in the external potential and is available up to arbitrary order (for actual calculations, however, it is advisable not to go beyond 4th order (the parameter settings of the implementation allow to run calculations up to about 10th order in the electron–nucleus potential).

The current implemtation is restricted to single-point calculations (gradients are not available) in C1-symmetry and cannot be used in parallel mode. Moreover, calculated properties using the DKH density do not take care of the 'picture-change' effect.

The arbitrary-order Hamiltonian requires in the control file:

`$dkhorder` *integer*

where *integer* specifies the order of the DKH Hamiltonian in the external potential, i.e. for the standard 2nd-order DKH Hamiltonian one should use `$dkhorder 2`.

The parametrization of the unitary transformation used in the DKH transformation can be optionally selected by

`$dkhparam` *integer*

The possible parametrizations in the DKH transformation are:

`$dkhparam 1:` Optimum parametrization (OPT)

`$dkhparam 2:` Exponential parametrization (EXP)

`$dkhparam 3:` Square-root parametrization (SQR)

`$dkhparam 4:` McWeeny parametrization (MCW)

`$dkhparam 5:` Cayley parametrization (CAY)

Note in particular that the parametrization does not affect the Hamiltonian up to fourth order. Therefore, as long as you run calculations with DKH Hamiltonians below 5th order you may use any symbol for the parametrization as they would all yield the same results. Higher-order DKH Hamiltonians depend slightly on the chosen paramterization of the unitary transformations applied in order to decouple the Dirac Hamiltonian, but this effect can be neglected.

For details on the arbitrary-order DKH Hamiltonians see Ref. [63] for details on the infinite-order DKH theory, [64] for the implementation, and [65] for a conceptual review of DKH theory. For details on the different parametrizations of the unitary transformations see [66].

6.6 Periodic Electrostatic Embedded Cluster Method

6.6.1 General Information

The Periodic Electrostatic Embedded Cluster Method (PEECM) functionality [67] provides electronic embedding of a finite, quantum mechanical cluster in a periodic, infinite array of point charges. It is implemented within HF and DFT energy and gradient `TURBOMOLE` modules: `dscf`, `grad`, `ridft`, `rdgrad`, and `escf`. Unlike embedding within a finite set of point charges the PEEC method always yields the correct electrostatic (Madelung) potential independent of the electrostatic moments of the point charges field. It is also significantly faster than the traditional finite point charges embedding.

6.6.2 Theoretical Background

Generally, the PEEC method divides the entire, periodic and infinite system into two parts, the inner (I) part, or so called cluster, and the outer (O) part which describes its environment. Thus, unlike "true" periodic quantum mechanical methods, PEECM primarily aims at calculations of structure and properties of localized defects in dominantly ionic crystals. The innermost part of the cluster is treated quantum mechanically (QM), whereas in the remaining cluster part cations are replaced by effective core potentials (ECPs) and anions by ECPs or by simply point charges. Such an "isolating" outer ECP shell surrounding the actual QM part is necessary in order to prevent artificial polarization of the electron density by cations which would otherwise be in a direct contact with the QM boundary. The outer part or the environment of the cluster is described by a periodic array of point charges, representing cationic and anionic sites of a perfect ionic crystal.

The electronic Coulomb energy term arising from the periodic field of point charges surrounding the cluster has the following form

$$J = \sum_{\mu\nu} \sum_{k}^{N\in \mathrm{UC}} \sum_{\vec{L}\in \mathrm{O}}^{\infty} D_{\mu\nu} q_k \int \frac{\mu(\vec{r})\nu(\vec{r})}{|\vec{r} - \vec{R}_k - \vec{L}|} d\vec{r},$$

where UC denotes the unit cell of point charges, $D_{\mu\nu}$ are elements of the density matrix, μ, ν are basis functions, q_k, $\vec{R}_k$ denote charges and positions of point charges, and $\vec{L}$ denote direct lattice vectors of the outer part O. It is evaluated using the periodic fast multipole method (PFMM) [68] which, unlike the Ewald method [69], defines the lattice sums entirely in the direct space. In general, PFMM yields a different electrostatic potential then the Ewald method, but the difference is merely a constant shift which depends on the shape of external infinite surface of the solid (i.e. on the way in which the lattice sum converges toward the infinite limit). However, this constant does not influence relative energies which are the same as obtained using the Ewald method, provided that the total charge of the cluster remains constant. Additionally, since the electrostatic potential within a solid is not

a well defined quantity, both the absolute total energies and orbital energies have no meaning (i.e. you cannot compare energies of neutral and charged clusters!).

6.6.3 Calculation Setup

There are three key steps in setting up a PEECM calculation. In the first step the periodic field of point charges has to be defined by specifying the point charges unit cell. Next step is the definition of the part infinite of point charges field that will be replaced by the explicit quantum mechanical cluster. Finally, the quantum mechanical cluster together with surrounding ECPs representing cationic sites as well as point charges representing anions is defined and put in place of the point charges. The input preparation steps can be summarized as follows

1. Dimensionality of the system is specified by the keyword `periodic` in the `$embed` section: `periodic 3` means a bulk three-dimensional system, `periodic 2` denotes a two-dimensional surface with an aperiodic z direction.

2. Definition of the unit cell of periodic point charges field is specified in the subsections `cell` and `content` of the `$embed` section.

3. Definition of the part of point charges field that will be replaced by the QM cluster together with the isolating shell (ECPs, explicit point charges) is specified in the subsection `cluster` of the `$embed` section.

4. Definition of the quantum mechanical cluster as well as the surrounding ECPs and anionic point charges is included in the usual `$coord` section.

The following two examples show the definition of the point charges unit cells.

Example 1. Ca_4F_{19} cluster embedded in bulk CaF_2

In this example a QM cluster with the composition Ca_4F_{19}, surrounded by 212 ECPs and 370 explicit point charges, representing Ca^{2+} cations and F^- anions is embedded in a periodic field of point charges (+2 for Ca and -1 for F) corresponding to the CaF_2 fluorite lattice.

First, the program has to know that this is a three-dimensional periodic system. This is specified by the keyword `periodic 3`, meaning periodicity in three dimensions. The dimensions of the unit cell for bulk CaF_2 are given in the subsection `cell` of the `$embed` keyword. By default, the unit cell dimensions are specified in atomic units and can be changed to Å using `cell ang`. The positions of the point charges in the unit cell are specified in the subsection `content`. In this example positions are given in fractional crystal coordinates (`content frac`). You can change this by specifying `content` for Cartesian coordinates in atomic units or `content ang` for Cartesian coordinates in Å. The values of point charges for Ca and F are given in the subsection `charges`.

```
$embed
```

```
periodic  3
cell
  10.47977 10.47977 10.47977 90.0 90.0 90.0
content frac
  F    0.00   0.00   0.00
  Ca  -0.25  -0.75  -0.75
  F    0.50  -0.50   0.00
  F    0.50   0.00  -0.50
  F    0.00  -0.50  -0.50
  F    0.50  -0.50  -0.50
  F    0.00   0.00  -0.50
  F    0.50   0.00   0.00
  F    0.00  -0.50   0.00
  Ca  -0.25  -0.25  -0.25
  Ca   0.25  -0.75  -0.25
  Ca   0.25  -0.25  -0.75
end
...
charges
  F   -1.0
  Ca   2.0
end
```

The above input defines a periodic, perfect, and infinite three-dimensional lattice of point charges corresponding to the bulk CaF_2 structure. In order to use this lattice for PEECM calculation we have to make "space" for our QM cluster and the isolating shell. This is done by specifying the part of the lattice that is virtually removed from the perfect periodic array of point charges to make space for the cluster. The positions of the removed point charges are specified in the subsection `cluster` of the `$embed` keyword. Note, that the position of the QM cluster and the isolating shell must **exactly** correspond to the removed part of the crystal, otherwise positions of the cluster atoms would overlap with positions of point charges in the periodic lattice, resulting in a "nuclear fusion".

```
cluster
  F    0.00000000000000   0.00000000000000   0.00000000000000
  Ca  -2.61994465796043  -2.61994465796043  -2.61994465796043
  Ca   2.61994465796043  -2.61994465796043   2.61994465796043
  Ca   2.61994465796043   2.61994465796043  -2.61994465796043
  Ca  -2.61994465796043   2.61994465796043   2.61994465796043
  F   -5.23988931592086   0.00000000000000   0.00000000000000
  F    0.00000000000000   0.00000000000000  -5.23988931592086
  F    5.23988931592086   0.00000000000000   0.00000000000000
  F    0.00000000000000  -5.23988931592086   0.00000000000000
```

```
  F    0.00000000000000   0.00000000000000   5.23988931592086
  F    0.00000000000000   5.23988931592086   0.00000000000000
  F   -5.23988931592086  -5.23988931592086   0.00000000000000
  F   -5.23988931592086   0.00000000000000  -5.23988931592086
  F   -5.23988931592086   0.00000000000000   5.23988931592086
  F   -5.23988931592086   5.23988931592086   0.00000000000000
  F    5.23988931592086  -5.23988931592086   0.00000000000000
  ...
```

repeated for $Ca_{216}F_{389}$

```
end
```

By default, the positions of point charges are specified in atomic units as Cartesian coordinates. You can change this by specifying `cluster frac` for fractional crystal coordinates or `cluster ang` for Cartesian coordinates in Å.

Finally, you have to specify the coordinates of the QM cluster along with the surrounding ECPs representing cationic sites and explicit point charges representing anions. This is done in the usual way using the `$coord` keyword.

```
$coord
    0.00000000000000      0.00000000000000      0.00000000000000      f
   -2.86167504097169     -2.86167504097169     -2.86167504097169      ca
    2.86167504097169      2.86167504097169     -2.86167504097169      ca
   -2.86167504097169      2.86167504097169      2.86167504097169      ca
    2.86167504097169     -2.86167504097169      2.86167504097169      ca
    0.00000000000000     -5.24009410923923      0.00000000000000      f
   -5.24009410923923      0.00000000000000      0.00000000000000      f
    0.00000000000000      5.24009410923923      0.00000000000000      f
    0.00000000000000      0.00000000000000     -5.24009410923923      f
    5.24009410923923      0.00000000000000      0.00000000000000      f
    0.00000000000000      0.00000000000000      5.24009410923923      f
    0.00000000000000     -5.24009410923923     -5.24009410923923      f
   -5.24009410923923     -5.24009410923923      0.00000000000000      f
    5.24009410923923     -5.24009410923923      0.00000000000000      f
    0.00000000000000     -5.24009410923923      5.24009410923923      f
    0.00000000000000      5.24009410923923     -5.24009410923923      f
 ...
```

repeated for $Ca_{216}F_{389}$

```
$end
```

This is the standard `TURBOMOLE` syntax for atomic coordinates. The actual distinction between QM cluster, ECP shell, and explicit point charges is made in the `$atoms` section.

```
$atoms
f  1,6-23                                                                \
   basis =f def-TZVP
ca 2-5                                                                   \
   basis =ca def-TZVP
ca 24-235                                                                \
   basis =none                                                           \
   ecp   =ca ecp-18 hay & wadt
f  236-605                                                               \
   basis =none                                                           \
   charge= -1.00000000
```

In the example above the F atoms 1 and 6-23 as well Ca atoms 2-5 are defined as QM atoms with def-TZVP basis sets. The Ca atoms 24-235 are pure ECPs and have no basis functions (`basis =none`) and F atoms 236-605 are explicit point charges with charge -1, with no basis functions and no ECP.

This step ends the input definition for the PEECM calculation.

Example 2. Al_8O_{12} cluster embedded in α-Al_2O_3 (0001) surface

In this example a QM cluster with the composition Al_8O_{12}, surrounded by 9 ECPs representing Al^{3+} cations is embedded in a two-dimensional periodic field of point charges (+3 for Al and -2 for O) corresponding to the (0001) surface of α-Al_2O_3.

As in the first example, the program has to know that this is a two-dimensional periodic system and this is specified by the keyword `periodic 2`. The dimensions of the unit cell for the (0001) α-Al_2O_3 surface are given in the subsection `cell` of the `$embed` keyword. The aperiodic direction is always the z direction, but you have to specify the unit cell as if it was a 3D periodic system. This means that the third dimension of the unit cell must be large enough to enclose the entire surface in this direction. The unit cell dimensions are specified in Å using `cell ang`. The positions of the point charges in the unit cell are specified as Cartesian coordinates in Å (`content ang`). The values of point charges for Al and O are given in the subsection `charges`.

```
$embed
periodic 2
cell angs
  4.8043    4.8043   24.0000   90.0000   90.0000  120.0000
content ang
  Al    2.402142286    1.386878848    5.918076515
  Al   -0.000013520   -0.000003382    7.611351967
  Al   -0.000008912    2.773757219    8.064809799
  Al    2.402041674    1.386946321    0.061230399
  Al   -0.000005568   -0.000003223   10.247499466
  Al    2.402137518    1.386872172    9.977437973
```

```
  Al    0.000000070    2.773757935    5.390023232
  Al    0.000006283   -0.000005607    3.696748018
  Al    2.402151346    1.386879444    3.243290186
  Al    0.000100868    2.773690462   11.246870041
  Al   -0.000001982   -0.000005796    1.060600400
  Al    0.000004853    2.773764610    1.330662251
  O    -0.731205344    1.496630311    6.749288559
  O     0.743527174    1.296469569    8.957922935
  O     1.588027477    0.104536049   11.127140045
  O     1.471626759    2.779079437    6.749288559
  O     3.309734344   -0.004341011    8.957920074
  O     3.919768333    1.323050499   11.127141953
  O    -0.740424335    4.045563698    6.749289513
  O    -1.651123047    2.868478537    8.957910538
  O     1.698525310    2.733071804   11.127161026
  O     3.133347750    2.664006472    4.558811665
  O     1.658615232    2.864167213    2.350177050
  O     0.814115047    4.056100845    0.180959582
  O     0.930515707    1.381557465    4.558811188
  O     1.494558096    0.004332162    2.350180149
  O    -1.517625928    2.837586403    0.180958077
  O     3.142566681    0.115072958    4.558810234
  O    -0.751034439    1.292158127    2.350189686
  O     0.703617156    1.427564979    0.180938885
end
...
charges
  O   -2.0
  Al   3.0
end
```

The above input defines a periodic, perfect, and infinite two-dimensional lattice of point charges corresponding to the (0001) α-Al_2O_3 surface. In order to use the lattice for PEECM calculation we have to make "space" for our QM cluster and the surrounding ECP shell. This is done by specifying the part of the lattice that is virtually removed from the perfect periodic array of point charges to make space for the cluster. The positions of the removed point charges are specified in the subsection `cluster` of the `$embed` keyword. Note, that the position of the QM cluster must **exactly** correspond to the removed part of the crystal, otherwise positions of the cluster atoms would overlap with positions of point charges in the periodic lattice, resulting in a "nuclear fusion".

```
cluster ang
  Al    -0.000012482    5.547518253    9.977437973
```

```
  Al     2.402141094    6.934402943    8.064809799
  Al     2.402144432    4.160642624   10.247499466
  Al     4.804287434    5.547518253    9.977437973
  Al     2.402250767    6.934336185   11.246870041
  Al    -0.000005568    8.321288109   10.247499466
  Al     2.402137518    9.708164215    9.977437973
  Al     4.804294586    8.321288109   10.247499466
  O      0.907584429    4.156304836    8.957920074
  O      1.517618299    5.483696461   11.127141953
  O     -0.703624666    6.893717766   11.127161026
  O      3.145677090    5.457115650    8.957922935
  O      3.990177393    4.265182018   11.127140045
  O      0.751026928    7.029124260    8.957910538
  O      4.100675106    6.893717766   11.127161026
  O      0.743527174    9.617761612    8.957922935
  O      1.588027477    8.425827980   11.127140045
  O      3.309734344    8.316950798    8.957920074
  O      3.919768333    9.644342422   11.127141953
  O      5.555326939    7.029124260    8.957910538
  Al     4.804400921   11.094982147   11.246870041
  Al    -0.000008912    2.773757219    8.064809799
  Al    -2.402049065    6.934336185   11.246870041
  Al     4.804400921    2.773690462   11.246870041
  Al     2.402136564    4.160642624    7.611351967
  Al    -0.000013520    8.321288109    7.611351967
  Al    -0.000008912   11.095048904    8.064809799
  Al     7.206440926    6.934402943    8.064809799
  Al     4.804286480    8.321288109    7.611351967
end
```

The positions of point charges are specified in Å as Cartesian coordinates.

Finally, you have to specify the coordinates of the QM cluster along with the surrounding ECPs. This is done in the usual way using the `$coord` keyword.

```
$coord
   -0.00002358760000     10.48329315900000     18.85463057110000      al
    4.53939007480000     13.10412613690000     15.24028611330000      al
    4.53939638280000      7.86247730390000     19.36497297520000      al
    9.07879006320000     10.48329315900000     18.85463057110000      al
    4.53959732680000     13.10399998250000     21.25351019750000      al
   -0.00001052200000     15.72496001430000     19.36497297520000      al
    4.53938331720000     18.34577677080000     18.85463057110000      al
    9.07880357850000     15.72496001430000     19.36497297520000      al
```

```
    1.71508649490000      7.85428007030000     16.92802041340000      o
    2.86788376470000     10.36268741690000     21.02725683720000      o
   -1.32965829240000     13.02724227310000     21.02729288000000      o
    5.94446987180000     10.31245694970000     16.92802581990000      o
    7.54034461170000      8.06002818410000     21.02725323160000      o
    1.41923561090000     13.28312353520000     16.92800239300000      o
    7.74915508620000     13.02724227310000     21.02729288000000      o
    1.40506312580000     18.17494056150000     16.92802581990000      o
    3.00093786570000     15.92251179600000     21.02725323160000      o
    6.25449323900000     15.71676368210000     16.92802041340000      o
    7.40729073370000     18.22517102690000     21.02725683720000      o
   10.49804944110000     13.28312353520000     16.92800239300000      o
    9.07900452260000     20.96648359440000     21.25351019750000      al
   -0.00001684120000      5.24164297480000     15.24028611330000      al
   -4.53921616520000     13.10399998250000     21.25351019750000      al
    9.07900452260000      5.24151682240000     21.25351019750000      al
    4.53938151440000      7.86247730390000     14.38337475740000      al
   -0.00002554910000     15.72496001430000     14.38337475740000      al
   -0.00001684120000     20.96660974680000     15.24028611330000      al
   13.61820356690000     13.10412613690000     15.24028611330000      al
    9.07878826040000     15.72496001430000     14.38337475740000      al
$end
```

This is the standard `TURBOMOLE` syntax for atomic coordinates. The actual distinction between QM cluster and ECP shell is made in the `$atoms` section.

```
$atoms
al 1-8                                                                 \
   basis =al def-SV(P)
o  9-20                                                                \
   basis =o def-SV(P)
al 21-29                                                               \
   basis =none                                                         \
   ecp   =al ecp-10 hay & wadt
```

In the example above the Al atoms 1-8 and O atoms 9-20 are defined as QM atoms with def-SV(P) basis sets. The Al atoms 21-29 are pure ECPs and have no basis functions (`basis =none`).

This step ends the input definition for the PEECM calculation.

6.7 Empirical Dispersion Correction for DFT Calculations

Based on an idea that has earlier been proposed for Hartree-Fock calculations [70,71], an general empirical dispersion correction has been proposed by Stefan Grimme for density functional calculations [72]. A modified version of the approach with extension to more elements and more functionals has been published in ref. [73].

The correction is invoked by the keyword `$disp` in the `control` file. The parameters of the second DFT-D publication are used. The older parameters are used when the keyword `$olddisp` is found in the `control` file.

When using the dispersion correction, the total energy is given by

$$E_{DFT-D} = E_{KS-DFT} + E_{disp} \tag{6.8}$$

where E_{KS-DFT} is the usual self-consistent Kohn-Sham energy as obtained from the chosen functional and E_{disp} is an empirical dispersion correction given by

$$E_{disp} = -s_6 \sum_{i=1}^{N_{at}-1} \sum_{j=i+1}^{N_{at}} \frac{C_6^{ij}}{R_{ij}^6} f_{dmp}(R_{ij})\,. \tag{6.9}$$

Here, N_{at} is the number of atoms in the system, C_6^{ij} denotes the dispersion coefficient for atom pair ij, s_6 is a global scaling factor that only depends on the DF used and R_{ij} is an interatomic distance. The interatomic C_6^{ij} term is calculated as geometric mean of the form

$$C_6^{ij} = \sqrt{C_6^i C_6^j}. \tag{6.10}$$

This yields much better results that the form used in the original paper:

$$C_6^{ij} = 2 \cdot \frac{C_6^i \cdot C_6^j}{C_6^i + C_6^j} \tag{6.11}$$

In order to avoid near-singularities for small R, a damping function f_{dmp} must be used which is given by

$$f_{dmp}(R_{ij}) = \frac{1}{1 + e^{-d(R_{ij}/R_r - 1)}} \tag{6.12}$$

where R_r is the sum of atomic vdW radii. These values are derived from the radius of the 0.01 a_0^{-3} electron density contour from ROHF/TZV computations of the atoms in the ground state. An earlier [72] used general scaling factor for the radii is decreased from 1.22 to 1.10 in the second implementation. This improves computed intermolecular distances especially for systems with heavier atoms. The atomic van der Waals radii R_0 used are given in Table 6.3 together with new atomic C_6 coefficients (see below). Compared to the original parameterization ($d = 23$), a smaller damping parameter of $d = 20$ provides larger corrections at intermediate distances (but still negligible dispersion energies for typical covalent bonding situations).

Table 6.2: s_6 parameters for functionals in the old and the revised implementation of DFT-D

Density Functional	s_6	s_6 (old)
BP86	1.05	1.30
B-LYP	1.20	1.40
PBE	0.75	0.70
B3-LYP	1.05	-[a]
TPSS	1.00	-[a]

[a] Not available [b] See Ref. [74]

Caution: if elements are present in the molecule for which no parameters are defined, the calculation proceeds with an atomic C_6 parameter of 0.0. This results in an incomplete description of the dispersion energy.

Table 6.3: C_6 parameters[a] (in Jnm^6mol^{-1}) and van der Waals radii[b] R_0 (in Å) for elements H-Xe.

element	C_6	R_0	element	C_6	R_0
H	0.14	1.001	K	10.80[c]	1.485
He	0.08	1.012	Ca	10.80[c]	1.474
Li	1.61	0.825	Sc-Zn	10.80[c]	1.562[d]
Be	1.61	1.408	Ga	16.99	1.650
B	3.13	1.485	Ge	17.10	1.727
C	1.75	1.452	As	16.37	1.760
N	1.23	1.397	Se	12.64	1.771
O	0.70	1.342	Br	12.47	1.749
F	0.75	1.287	Kr	12.01	1.727
Ne	0.63	1.243	Rb	24.67[c]	1.628
Na	5.71[c]	1.144	Sr	24.67[c]	1.606
Mg	5.71[c]	1.364	Y-Cd	24.67[c]	1.639[d]
Al	10.79	1.639	In	37.32	1.672
Si	9.23	1.716	Sn	38.71	1.804
P	7.84	1.705	Sb	38.44	1.881
S	5.57	1.683	Te	31.74	1.892
Cl	5.07	1.639	I	31.50	1.892
Ar	4.61	1.595	Xe	29.99	1.881

[a] Derived from UDFT-PBE0/QZVP computations. [b] Derived from atomic ROHF/TZV computations. [c] Average of preceeding group VIII and following group III element. [d] Average of preceeding group II and following group III element.

Table 6.4: old C_6 parameters[a] (in Jnm^6mol^{-1}) and van der Waals radii[b] R_0 (in Å) for elements H-Ne.

element	C_6	R_0	element	C_6	R_0
H	0.16	1.11	O	0.70	1.49
C	1.65	1.61	F	0.57	1.43
N	1.11	1.55	Ne	0.45	1.38

[a] Derived from UDFT-PBE0/QZVP computations. [b] Derived from atomic ROHF/TZV computations.

Chapter 7

Hartree–Fock and DFT Response Calculations: Stability, Dynamic Response Properties, and Excited States

7.1 Functionalities of ESCF and EGRAD

`escf` and `egrad` are designed as efficient tools for response and excited state calculations on large molecules. `escf` serves to compute the following properties for HF and KS reference states:

- Eigenvalues of the electronic Hessian (stability analysis)
- Frequency-dependent polarizabilities and optical rotations
- Vertical electronic excitation energies
- Transition moments, oscillator and rotatory strengths of electronic excitations ⇒ UV-VIS and CD spectra

Spin-restricted closed-shell and spin-unrestricted ground states (except for stability analysis) are supported. The RI-J approximation in conjunction with LDA and GGA functionals is implemented for all properties. Excitation energies and transition moments can be computed either within the full time-dependent HF (TDHF) or time-dependent DFT (TDDFT) formalisms or within the Tamm-Dancoff approximation (TDA).

Excited state first order properties can be evaluated analytically using `egrad`. They include:

- Gradients of the excited state energy with respect to nuclear positions
  ⇒ Excited state equilibrium structures (`jobex`), adiabatic excitation energies, emission spectra
- Exited state densities ⇒ Charge moments, population analysis
- Excited state force constants by numerical differentiation of gradients (using the script `Numforce`)

Moreover, analytical gradients of static and frequency-dependent polarizabilities are available from `egrad`. Together with vibrational normal modes from the `aoforce` or `Numforce` they are used to calculate vibrational Raman intensities.

Again, ground states may be spin-restricted closed-shell or spin-unrestricted, RI-J is available, and either full TDDFT/TDHF or the TDA can be used. For further details we refer to a recent review [75].

7.2 Theoretical Background

We briefly state the basic working equations in the following, as far as required to understand the program output. For a more detailed treatment of the theory see refs. [76, 16, 75, 77, 78] and refs. therein.

The first-order frequency dependent response of the density matrix can be expanded as

$$\gamma(x,x') = \sum_{ai} \{X_{ai}\varphi_i(x)\varphi_a^*(x') + Y_{ai}\varphi_a(x)\varphi_i^*(x')\}. \tag{7.1}$$

The (real) expansion coefficients X_{ai} and Y_{ai} are conveniently gathered in a "super-vector"

$$|X,Y\rangle = \begin{pmatrix} X \\ Y \end{pmatrix} \tag{7.2}$$

on L, the linear space of products of occupied and virtual ground state MOs $\varphi_i(x)\varphi_a^*(x')$ plus their complex conjugates. X and Y describe the first-order change of the ground state MOs due to an external perturbation which is represented by $|P,Q\rangle$ on L. For example, if an oscillating electric dipole perturbation along the z axis is applied, $|P,Q\rangle = |\mu_z\rangle$, where μ is the electric dipole operator.

Next we define the 2×2 "super-matrices"

$$\Lambda = \begin{pmatrix} A & B \\ B & A \end{pmatrix}, \quad \Delta = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{7.3}$$

where the four-index quantities A and B are the so-called "orbital rotation Hessians". Explicit expressions for A and B can be found, e.g., in ref. [16]. The vector $|X,Y\rangle$ is determined as the solution of the TDHF/TDDFT response problem,

$$(\Lambda - \omega\Delta)|X,Y\rangle = -|P,Q\rangle. \tag{7.4}$$

If $|X_\alpha, Y_\alpha\rangle$ arises from an electric dipole perturbation $|\mu_\alpha\rangle$, the electronic dipole polarizability at frequency ω is

$$\alpha_{\alpha\beta}(\omega) = -\langle X_\alpha, Y_\alpha|\mu_\beta\rangle, \tag{7.5}$$

$\alpha, \beta \in \{x, y, z\}$. Similarly, if $|m_\alpha\rangle$ is a component of the magnetic dipole moment operator, the optical rotation is [79]

$$\delta_{\alpha\beta}(\omega) = -\tfrac{c}{3\omega}\mathrm{Im}\langle X_\alpha, Y_\alpha|m_\beta\rangle, \tag{7.6}$$

where c is the light velocity.

Excitation energies Ω_n are the poles of the frequency-dependent density matrix response. They are thus the zeros of the operator on the left-hand side of Eq. (7.4),

$$(\Lambda - \Omega_n\Delta)|X_n, Y_n\rangle = 0. \tag{7.7}$$

The corresponding eigenvectors $|X_n, Y_n\rangle$ are the transition density matrices for a given excitation (also called "excitation vectors" in the following). They are required to be normalized according to

$$\langle X_n, Y_n|\Delta|X_n, Y_n\rangle = 1. \tag{7.8}$$

Transition moments are evaluated by taking the trace with one-particle operators, e.g.,

$$\boldsymbol{\mu}^{0n} = \langle X_n, Y_n|\boldsymbol{\mu}\rangle \tag{7.9}$$

for the electric and

$$\mathbf{m}^{0n} = \langle X_n, Y_n|\mathbf{m}\rangle \tag{7.10}$$

for the magnetic transition dipole moments.

The full TDHF/TDDFT formalism is gauge-invariant, i.e., the dipole-length and dipole-velocity gauges lead to the same transition dipole moments in the basis set limit. This can be used as a check for basis set quality in excited state calculations. The TDA can formally be derived as an approximation to full TDHF/TDDFT by constraining the Y vectors to zero. For TDHF, the TDA is equivalent to configuration interaction including all single excitations from the HF reference (CIS). The TDA is not gauge invariant and does not satisfy the usual sum rules [76], but it is somewhat less affected by stability problems (see below).

Stability analysis of closed-shell electronic wavefunctions amounts to computing the lowest eigenvalues of the electric orbital rotation Hessian $A + B$, which decomposes into a singlet and a triplet part, and of the magnetic orbital rotation Hessian $A - B$. Note that $A - B$ is diagonal for non-hybrid DFT, while $A + B$ generally is not. See refs. [80, 15] for further details.

Properties of excited states are defined as derivatives of the excited state energy with respect to an external perturbation. It is advantageous to consider a fully variational Lagrangian of the excited state energy [16],

$$\begin{aligned} L[X, Y, \Omega, C, Z, W] = E_{\mathrm{GS}} &+ \langle X, Y|\Lambda|X, Y\rangle - \Omega\big(\langle X, Y|\Delta|X, Y\rangle - 1\big) \\ &+ \sum_{ia} Z_{ia}F_{ia} - \sum_{pq} W_{pq}(S_{pq} - \delta_{pq}). \end{aligned} \tag{7.11}$$

Here E_{GS} denotes the ground state energy, F and S are the Fock and overlap matrices, respectively, and indices p, q run over all, occupied and virtual MOs.

First, L is made stationary with respect to *all* its parameters. The additional Lagrange multipliers Z and W enforce that the MOs satisfy the ground state HF/KS equations and are orthonormal. Z is the so-called Z-vector, while W turns out to be the excited state energy-weighted density matrix. Computation of Z and W requires the solution of a *single* static TDHF/TDKS response equation (7.4), also called coupled and perturbed HF/KS equation. Once the relaxed densities have been computed, excited state properties are obtained by simple contraction with derivative integrals in the atomic orbital (AO) basis. Thus, computation of excited state gradients is more expensive than that of ground state gradients only by a constant factor which is usually in the range of $1 \ldots 4$.

TDHF/TDDFT expressions for components of the frequency-dependent polarizability $\alpha_{\alpha\beta}(\omega)$ can also be reformulated as variational polarizability Lagrangians [81]

$$\begin{aligned} & L^{\alpha\beta}[X_\alpha, Y_\alpha, X_\beta, Y_\beta, C, Z^{\alpha\beta}, W^{\alpha\beta}](\omega) \\ & \quad = \langle X_\alpha, Y_\alpha | (\Lambda - \omega\Delta) | X_\beta, Y_\beta \rangle + \langle X_\alpha, Y_\alpha | \mu_\beta \rangle + \langle \mu_\alpha | X_\beta, Y_\beta \rangle \\ & \quad + \sum_{ia\sigma} Z^{\alpha\beta}_{ia\sigma} F_{ia\sigma} - \sum_{pq\sigma, p \le q} W^{\alpha\beta}_{pq\sigma} (S_{pq\sigma} - \delta_{pq}). \end{aligned} \tag{7.12}$$

The stationary point of $L^{\alpha\beta}(\omega)$ equals to $-\alpha_{\alpha\beta}(\omega)$. The requirement that $L^{\alpha\beta}(\omega)$ be stationary with respect to all variational parameters determines the Lagrange multipliers $Z^{\alpha\beta}$ and $W^{\alpha\beta}$. All polarizability components $\alpha\beta$ are processed simultaneously which allows for computation of polarizability derivatives at the computational cost which is only 2–3 higher than for the electronic polarizability itself.

7.3 Implementation

Without giving details, we discuss features of the implementation in `escf` and `egrad` that matter for applications. The interested reader is referred to the refs. given in the program headers as well as ref. [82].

Simultaneous vector iteration. The solutions of Eqs. (7.4) and (7.7) are expanded in a subspace of L which is iteratively expanded (Davidson method [83]). The iteration is stopped when the Euclidean norm of the residual vector is smaller than 10^{-k}. The default for k is 5, which usually gives excitation energies accurate to $8-10$ digits and properties accurate to $4-5$ digits (k can be changed by specifying `$rpaconv` k). Several roots, i.e., several excited states or frequencies can be treated simultaneously, which is very effective and permits the calculation of whole excitation spectra and dispersion curves. During the iteration, the vectors are kept on scratch files `vfile_<IR>`,`wfile_<IR>`, and/or `rhs_<IR>`, where `IR` denotes an IREP of the point group (see below). Before the programs terminate, the converged vectors are written onto formatted files ⟨type⟩⟨IR⟩, where type is an abbreviation

for the type of response calculation performed (cf. `$scfinstab`). Given these files in the working directory, `escf` and `egrad` calculations can be restarted or continued, e.g., with a larger number of roots.

Integral direct algorithm. In the iterative method outlined above, the supermatrices A and B never need to be set up explicitly; only the products of A and B with some suitable basis vectors are required. These matrix-vector-products are evaluated very efficiently in the AO basis, because the required four-index integrals can be computed "on the fly" and need not be transformed or stored on disk. In addition, prescreening techniques based on rigorous bounds are straightforward to apply. This leads to a low-order scaling $O(N^2) - O(N)$ for the time-determining steps. Due to the similarity to ground state fock matrix construction, the same keywords are used to control these steps as in semi-direct SCF, namely `$thime`, `$thize`, `$scfintunit`, see Chapter 6. The same is true for DFT and RI keywords such as `$dft`, `$ridft`, `$ricore`.

Point group symmetry. `escf` and `egrad` can exploit point group symmetry for *all* finite point groups (with up to 99-fold symmetry axes, → `$symmetry`). The response and eigenvalue problems (7.4) and (7.7) decompose into separate problems for each IRREP that are solved independently. For excited state and instability calculations, it is thus necessary to specify the IRREPs to be treated (`$soes`, see below). For response calculations, the perturbation is automatically subduced into irreducible components. The overall speedup compared to C_1 symmetry is approximately $1/g$, where g denotes the point group order. For spin-restricted closed-shell ground states, spin symmetry is used to further reduce the dimension of the response and eigenvalue problems by a factor of 2.

Other features. `escf` and `egrad` fully support external fields (using the keyword `$electrostatic field`; specify `geofield on` in `$fldopt`), point charges (using the keyword `$point_charges`), and effective core potentials (using `$ecp`). In `escf` calculations, occupied and virtual MOs can be frozen (using `$freeze`).

7.4 How to Perform

The most convenient way to set up an `escf` or `egrad` calculation is to use the `ex` option of the last ("general") `define` menu, see Chapter 4. `define` will automatically provide most of the keywords discussed below.

A large number of (not necessarily realistic) sample inputs is contained in the `escf` and `egrad` subdirectories of the test suite (`TURBOTEST` directory).

7.4.1 Preliminaries

All response calculations require a complete set of converged (occupied and virtual) SCF MOs. It is strongly recommended to use *well converged MOs*, since the error in the ground-state wavefunction enters linearly in all response properties. Thus, before starting `escf` or `egrad`, specify the keywords

```
$scfconv 7
$denconv 1d-7
```

in `control`, perform a `dscf` statistics run, if semi-direct integral processing is to be used (see Chapter 3.1), and (re-)run `dscf` or `ridft`,

`dscf > dscf.out &` or

`ridft > ridft.out &` in case of RI-*J*.

The above tight convergence criteria are also recommended for excited state geometry optimizations.

7.4.2 Polarizabilities and Optical Rotations

The calculation of dynamic polarizabilities is controlled by the keyword

`$scfinstab dynpol` *unit*
list of frequencies

unit specifies the unit of the following frequencies and may be `ev`, `nm`, `1/cm`, or `a.u.` (default). The frequencies may be either purely real or purely imaginary. For example, to calculate dynamic polarizabilities at 590 nm and 400 i nm (i is the imaginary unit), specify

```
$scfinstab dynpol nm
  590
  400 i
```

and run `escf`,

`escf > escf.out &` .

The resulting polarizabilities and rotatory dispersions are given in a.u. in the program output (`escf.out` in the above example). For conversion of the optical rotation in a.u. to the specific rotation in deg·[dm·(g/cc)]$^{-1}$ see Eq. (15) of ref. [79].

Note that convergence problems may occur if a frequency is close to an electronic excitation energy. This is a consequence of the (physical) fact that the response diverges at the excitation energies, and not a problem of the algorithm.

Static polarizabilities are calculated most efficiently by specifying

```
$scfinstab polly
```

before starting `escf`.

7.4.3 Stability Analysis

Stability analysis of spin-restricted closed-shell ground states is enabled by

`$scfinstab singlet`
: for singlet instabilities,

`$scfinstab triplet`
: for triplet instabilities (most common), and

`$scfinstab non-real`
: for non-real instabilities.

After that, it is necessary to specify the IRREPs of the electronic Hessian eigenvectors ("orbital rotations") to be considered. Without additional knowledge of the system one usually needs to calculate the lowest eigenvalue within every IRREP:

```
$soes all 1
```

Positivity of the lowest eigenvalues in all IRREPs is sufficient for stability of the ground state solution. If one is interested in, say, the lowest eigenvalues in IRREPs `eg` and `t2g` only, one may specify:

```
$soes
eg   1
t2g  1
```

Triplet instabilities in the totally symmetric IRREP indicate open shell diradical states (singlet or triplet). In this case, start MOs for spin-symmetry broken UHF or UKS ground state calculation can be generated by specifying

```
$start vector generation
```

`escf` will provide the start MOs (→ `$uhfmo_alpha`, `$uhfmo_beta`) as well as occupation numbers (→ `$alpha shells`, `$beta shells`) for a spin-unrestricted calculation with equal numbers of α and β electrons (pseudo-singlet occupation).

7.4.4 Vertical Excitation and CD Spectra

The calculation of excited states within the TDHF(RPA)/TDDFT approach is enabled by

`$scfinstab rpas`
for closed-shell singlet excitations,

`$scfinstab rpat`
for closed-shell triplet excitations, and

`$scfinstab urpa`
for excitations out of spin-unrestricted reference states.

If it is intended to use the TDA instead, specify

`$scfinstab ciss`
for closed-shell singlet excitations,

`$scfinstab cist`
for closed-shell triplet excitations, and

`$scfinstab ucis`
for excitations out of spin-unrestricted reference states.

Next, the IRREPs of the excitations need to be defined, which is again accomplished using `$soes`. For example, to calculate the 17 lowest excitations in IRREP b1g, the 23 lowest excitations in IRREP eu, and all excitations in IRREP t2g, use

```
$soes
b1g  17
eu   23
t2g  all
```

and run `escf`.

Note that `$soes` specifies the IRREP of the *excitation vector* which is not necessarily identical to the IRREP of the *excited state(s)* involved. In general, the IRREP(s) of the excitation(s) from the ground to an excited state is given by the direct product of the IRREPs of the tow states. For example, to calculate the first A_2 state in a C_{2v}-symmetric molecule with a B_2 (open-shell) ground state, it is necessary to specify

```
$soes
b1   1
```

The number of excitations that have to be calculated in order to cover a certain spectral range is often difficult to determine in advance. The total number of excitations within each IRREP as provided by the `define ex` menu may give some hint. A good strategy is to start with a smaller number of excitations and, if necessary, perform a second `escf` run on a larger number of states using the already converged excitation vectors as input.

To compute absorption and CD spectra, it is often sufficient to include optically allowed transitions only. This leads to substantial reduction of computational effort for molecules with higher symmetry. For example, in the UV-VIS spectrum of an O_h symmetric molecule, only t_{1u} excitations are optically allowed. The IRREPs of the electric and magnetic dipole moments as well as of the electric quadrupole moment are displayed automatically in the `define ex` menu.

If a large number of states is to be calculated, it is highly recommended to provide extra memory by specifying

`$rpacor` *m*

the integer *m* being the core memory size in megabytes (default is 20). The larger *m*, the more vectors can be processed simultaneously without re-calculation of integrals. As a rule of thumb, *m* should be ca. 90% of the available main memory. If RI-*J* is used (`$ridft`), it is recommended to set `$ricore` to a small value and `$rpacor` to a large value if the number of states is large, and vice versa if it is small.

By specifying

`$spectrum` *unit* and/or

`$cdspectrum` *unit*

a list of excitation energies and oscillator and/or rotatory strengths of the optically allowed transitions is written onto file `spectrum` and/or `cdspectrum`. As above, unit specifies the energy unit and may be `ev`, `nm`, `1/cm`, or `a.u.` (default). The files `spectrum` and `cdspectrum` may conveniently be used for further processing, e.g., using a plotting program such as Gnuplot.

7.4.5 Excited State Geometry Optimizations

The input for computing excited state gradients and properties using `egrad` is exactly the same as for an excited state calculation using `escf`, see the previous section. Gradients and properties are calculated only for one state at a time. By default, this is the highest excitation specified by `$soes` (only one IRREP is allowed). Sometimes, e.g. close to excited state intersections, it may be necessary to include higher excited states in the initial excitation vector calculation to prevent root flipping. This is accomplished using

`$exopt` *n*

which explicitly enforces treatment of the *n*-th state; *n* must be less or equal the number of states specified in `$soes`.

After the input for the ground and excited state calculations has been set up, an excited state geometry optimization can be started by issuing the command

```
nohup jobex -ex &
```

The option `-ex` forces `jobex` to call `egrad` instead of `grad` (or `rdgrad` if `-ri` is also specified). In each geometry step, the excitation energy is written on the fourth column in `$energy`, and the data group `$last excitation energy change` is updated. Otherwise, the excited state optimization proceeds in exactly the same way as a ground state optimization (see Chapter 3.1).

7.4.6 Excited State Force Constant Calculations

Excited state vibrational frequencies can be calculated by numerical differentiation of analytic gradients using `Numforce` (see Chapter 10). A `Numforce` calculation for an excited state may be started by the command

`nohup NumForce -ex` n `> force.out &`

where n is the number of the excited state *in* C_1 *symmetry*. In order to determine n, it is recommended to perform an `escf` calculation in C_1 symmetry. Note that numerical calculation of excited state force constants *is likely to fail* if there are other states nearby (in C_1), because the roots may flip when the molecule is distorted. Note also that it may be necessary to include higher excited states (using `$exopt`, see above) in C_1 calculations of molecules with higher symmetry in order to enforce convergence to the correct state. In any case, it should be checked that the energy change due to the displacements (available in the `numforce/KraftWerk/*.log` files) is reasonably small.

For a `Numforce` run, the convergence criteria should be tightened. It is recommended to use at least

```
$scfconv 8
```

in all `Numforce` calculations. Other `Numforce` options such as `-central`, `-d`, `-np` work in exactly the same way as they do for ground states.

7.4.7 Polarizability Derivatives and Raman Spectra

Calculations of polarizability derivatives by the `egrad` program use the same specifications in the `$scfinstab` data group as polarizability calculations by `escf`.

```
$scfinstab polly
```

specifies derivatives of the static polarizability, while

`$scfinstab dynpol` *unit*
frequency

requests derivatives of the dynamical polarizability at the given frequency. Note that, unlike polarizability calculations, multiple frequencies are not allowed. Polarizability derivatives have to be projected onto vibrational normal modes to obtain Raman intensities, see Chapter 10 for further details.

Chapter 8

Second-order Møller–Plesset Perturbation Theory

8.1 Functionalities of MPGRAD, RIMP2, and RICC2

TURBOMOLE offers three possibilities for the calculation of MP2 data. A "conventional" implementation [84], `mpgrad`, based on the calculation of four-center integrals (not further developed for several years), and two treatments within the resolution-of-the-identity (RI) approximation: the first and stand–alone implementation [8], `rimp2`, and within the coupled–cluster program [10], `ricc2`.

Functionality of `mpgrad`:

- Calculation of MP2 energies and/or MP2 gradients for RHF and UHF wave functions.
- The frozen core approximation (possibility to exclude low-lying orbitals from the MP2 treatment) is implemented only for MP2 energies.
- Exploitation of symmetry of all point groups.

Functionality of `rimp2`:

- Calculation of MP2 energies and/or gradients for RHF and UHF wave functions within the efficient RI-approximation.
- The frozen core approximation is implemented for both RI-MP2 energies and gradients.
- RI-MP2 needs optimised auxiliary basis sets, which are available for all TURBOMOLE standard basis sets (SVP, TZVP, TZVPP, QZVPP) as well as for the (aug-)cc-p(wC)VXZ (X = D, T, Q, 5) basis sets series (for Al–Ar also for the (aug-)cc-p(wC)V(X+d)Z series).

- Exploitation of symmetry of all point groups.

Functionality of `ricc2`:

- Includes the above `rimp2` functionalities.
- Runs parallel.
- See Chapter 9 for further details.

8.2 Some Theory

Second-order Møller–Plesset Perturbation Theory (MP2) corrects errors introduced by the mean-field ansatz of the Hartree–Fock (HF) theory, the perturbation operator is just the difference of the exact and the HF Hamiltonian. One straightforward obtains the MP2 energy:

$$E_{MP2} = \frac{1}{4} \sum_{iajb} \left[t_{ij}^{ab} \langle ij||ab\rangle \right], \tag{8.1}$$

with the t-amplitudes

$$t_{ij}^{ab} = \frac{\langle ij||ab\rangle}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}, \tag{8.2}$$

i and j denote occupied, a and b virtual orbitals, ϵ_p are the corresponding orbital energies, $\langle ij||ab\rangle = \langle ij|ab\rangle - \langle ij|ba\rangle$ are four-center-two-electron integrals in a commonly used notation.

MP2 gradients (necessary for optimisation of structure parameters at the MP2 level) are calculated as analytical derivatives of the MP2 energy with respect to nuclear coordinates; calulation of these derivatives also yields the first order perturbed wave function, expressed as "MP2 density matrix", in analogy to the HF density matrix. MP2 corrections of properties like electric moments or atomic populations are obtained in the same way as for the HF level, the HF density matrix is just replaced by the MP2 density matrix.
The "resolution of the identity (RI) approximation" means expansion of products of virtual and occupied orbitals by expansions of so-called "auxiliary functions". Calculation and transformation of four-center-two-electron integrals (see above) is replaced by that of three-center integrals, which leads to computational savings of `rimp2` (compared to `mpgrad`) by a factor of ca. 5 (small basis sets like SVP) to ca. 10 (large basis sets like TZVPP) or more (for cc-pVQZ basis sets). The errors (differences to `mpgrad`) of `rimp2` in connection with optimised auxliliary basis sets are small and well documented [9, 85]. The use of the `mpgrad` modul is recommended rather for reference calculations or if suitable auxiliary basis sets are not available.

8.3 How to Prepare and Perform MP2 Calculations

Prerequisites

Calculations with `mpgrad`, `rimp2` or `ricc2` require

- a converged SCF calculation with the one-electron density convergence threshold set to `$denconv 1.d-7` or less
- the maximum core memory the program is allowed to allocate should be defined in the data group `$maxcor` (in MB); the recommended value is ca. 3/4 of the available (physical) core memory at most.
- orbitals to be excluded from the correlation treatment have to be specified in data group `$freeze`
- the calculation of MP2 gradients is omitted by adding the flag `$mp2energy` to the `control` file; in this case only MP2 energy is calculated.

Calculations with `rimp2` moreover require

- an auxiliary basis defined in the data group `$cbas`

this is not needed for `mpgrad` , but here one needs

- a specification for scratch files and their size in data group `$mointunit` (see Section 14.2.12)
- and the number of passes for integral evaluations and transformations in data group `$traloop`

Rimp2 Calculations

1. RI-MP2 calculations require the specification of auxiliary basis sets (`$cbas`) and a converged SCF calculation with the one-electron density convergence threshold set to `$denconv 1.d-7` or less. In addition, the options `$freeze` (frozen core approximation) and `$maxcor` (maximum core memory usage) should be set. All these settings can be done during the input generation with the program `define` under the entry `mp2\cc2` of last main menu.
2. Alternatively, the interactive program `Rimp2prep` can be used: This program sets default values for auxiliary basis sets (data group `$cbas`), for frozen core orbitals (data group `$freeze`, all orbitals with energies below -3 a.u. are suggested to be frozen) and for the amount of memory to be allocated (`$maxcor`. These defaults can be confirmed with `return` or modified, if desired. Note: the amount of memory to be allocated determines the number of multiple passes and thus the efficiency of `rimp2`.
   It is also possible to run `Rimp2prep` directly after `define`.

3. The `ricc2` program requires the data group:
   ```
   $ricc2
     mp2
     geoopt model=mp2
   ```
   This can be prepared with `define` in the menu `mp2\cc2`.

4. Start a single `rimp2` calculation with the command `ricc2` or `rimp2`.

5. For optimisation of structure parameters at the RI-MP2 level use the command `jobex -level cc2` or `jobex -ri -level mp2`.

Mpgrad Calculations

1. Add `$denconv 1.d-7` to the `control` file and perform a `dscf` run.

2. If any orbitals are decided to be excluded from MP2 treatment, add data group `$freeze` manually to the `control` file, see also Section 14.2.12.

3. For preparation of an `mpgrad` run use the script `Mp2prep`:
   `mp2prep -e/g -m` *memory* `-p` *discspace [scratch file directory]*
   As an example, with the command
   `mp2prep -e -m 100 -p 1000 /work`
   an MP2-energy calculation is prepared, the amount of available core memory is restricted to 100 MB, the MOs are blocked, so that integral scratch files—located in the directory /work—do not need more than 1000 MB. The number of blocks, i.e. the number of passes with repeated integral evaluations, is written to the `control` file (`$traloop`) as well as the specification of scratch files (`$mointunit`, see Section 14.2.12). Note: less disc space means more passes and thus lower efficiency of `mpgrad`. Settings obtained by `Mp2prep` may be changed manually. You may change the number of passes in `$traloop` by editing the `control` file (e.g. if the originally intended disc space is not available). To adapt the size of scratch files add `$statistics mpgrad` to `control` file and start an `mpgrad` statistics run with the command `mpgrad`.

4. Start a single `mpgrad` calculation with the command `mpgrad`.

5. For optimisation of structure parameters at the (non-RI-) MP2 level use the command `jobex -level mp2`. Note, that the frozen core approximation is ignored in this case.

8.4 General Comments on MP2 Calculations, Practical Hints

Recommendations

- It is well-known, that perturbation theory yields reliable results only, if the perturbation is small. This is also valid for MP2, which means, that MP2 improves HF results only, if HF already provides a fairly good solution to the problem. If HF fails, e.g. in case of partially filled *d*-shells, MP2 usually will also fail and should not be used in this case.
- MP2 results are known to converge very slowly with increasing basis sets, in particular slowly with increasing *l*-quantum number of the basis set expansion. Thus for reliable results the use of TZVPP basis sets (or higher) is recommended. When using SVP basis sets a qualitative trend can be expected at the most. Basis sets much larger than TZVPP usually do not significantly improve geometries of bonded systems, but still can improve the energetic description. For non–bonded systems larger basis sets (especially, with more diffuse functions) are needed.
- It is recommended to exclude all non-valence orbitals from MP2 calculations, as neither `TURBOMOLE` standard basis sets nor cc-pVXZ (X=T,Q,5) are designed for correlation treatment of inner shells (for this purpose polarisation functions for the inner shells are needed). The default selection for frozen core orbitals in `Define` (orbitals below -3 a.u. are frozen) provides a reasonable guess.
- RI-MP2: We strongly recommend the use of auxiliary basis sets optimized for the corresponding (MO) basis sets.

Comments on Output

- Most important output for `ricc2`, `rimp2`, and `mpgrad` are of course MP2(+HF) energies (written standard output and additionally to file `energy`) and MP2(+HF) gradients (written to file `gradient`).
- In case of MP2 gradient calculations the modules also calculate the MP2 dipole moment from the MP2 density matrix (note, that in case of `mpgrad` frozen core orbital specification is ignored for gradient calculations and thus for MP2 dipole moments).

Further output contains indications of the suitability of the (HF+MP2) treatment.

- As discussed above, reliable (HF+MP2) results are in line with small MP2 corrections. The size of the MP2 correction is characterised by the t-amplitudes, as evident from the above equations. `mpgrad` by default plots the five largest t-amplitudes as well as the five largest norms of t-amplitudes for fixed i and j,

`rimp2` does the same upon request, if `$tplot` is added to `control` file. More or less than five t-amplitudes will be plotted for `$tplot` n, where n denotes the number of largest amplitudes to be plotted. It is up to the user to decide from these quantities, whether the (SCF+MP2) treatment is suited for the present problem or not. Unfortunately, it is not possible to define a threshold, which distinguishes a "good" and a "bad" MP2-case, but comparison of values for several similar calulations will yield some information. Example: the largest norm of t-amplitudes for the Cu-atom ($d^{10}s^1$, "good" MP2-case) amounts to ca. 0.06, that of the Ni-atom (d^8s^2, "bad" MP2 case) is ca. 0.14.

- A more descriptive criterion may be derived from the MP2 density matrix. The eigenvalues of this matrix reflect the changes in occupation numbers resulting from the MP2 treatment, compared to the SCF density matrix, where occupation numbers are either one (two for RHF) or zero. Small changes mean small corrections to HF and thus suitability of the (HF+MP2) method for the given problem. In case of gradient calculations `rimp2` displays by default the largest eigenvalue of the MP2 density matrix, i.e. the largest change in occupation numbers (in %). All eigenvalues are shown, if `$mp2occ` is added to the `control` file. For main group compounds largest changes in occupation numbers of ca. 5 % or less are typical, for d^{10} metal compounds somewhat higher values are tolerable.

Chapter 9

Second-Order Approximate Coupled-Cluster (CC2) Calculations

`ricc2` is a module for the calculation of excitation energies and response properties at a correlated *ab initio* level, in particular the second-order approximate coupled-cluster model CC2 [86]. All calculations employ the resolution-of-the-identity (RI) approximation for the electron repulsion integrals needed for the correlation treatment and and the description of excitation processes. At present the following functionalities are implemented:

ground state energies for MP2 and CC2 and spin-component scaled variants thereof; the MP2 results are identical with those obtained with `rimp2` (but usually the calculations are somewhat faster).

excitation energies for the models CIS/CCS, CIS(D), CIS(D_∞), ADC(2), and CC2

transition moments for ground state—excited and excited—excited state transitions for the models CCS and CC2

first-order properties for the ground state (SCF/CCS, MP2, and CC2) and excited states (CCS, CC2, ADC(2) and CIS(D_∞))

geometric gradients for the electronic ground state at the MP2 and the CC2 level; for electronically excited states at the CIS(D_∞), ADC(2), and CC2 level

gradients for auxiliary basis sets for RI-MP2, -CC2, etc. calculations based on the RI-MP2 error functional

F12 corrections to RI-MP2; MP2 ground-state energies can be computed (in C_1 symmetry) using explicitly-correlated two-electron basis functions in the framework of the MP2-F12 model [87, 88].

All functionalities are implemented for closed-shell RHF and open-shell UHF reference wavefunctions. Ground (MP2, CC2) and excited state energies (CC2, CIS(D_∞), ADC(2)) are also implemented for single determinant restricted open-shell reference wavefunctions.

Prerequisites

Calculations with the `ricc2` module require (almost) the same prerequisites as RI-MP2 calculations:

1. a converged SCF calculation with the one-electron density convergence threshold set to `$denconv 1.d-5` or less
2. an auxiliary basis defined in the data group `$cbas`
3. if orbitals should be excluded from the correlation treatment (and excitation processes) the data group `$freeze` has to be set
4. the maximum core memory which the program is allowed to allocate should be defined in the data group `$maxcor`; the recommended value is 66–75% of the available (physical) core memory.
5. depending on the type of calculations that should be carried out, additionally the data groups `$ricc2`, `$excitations`, `$response`, and `$rir12` have to be set (see below and Section 14.2.13).

For calculations with the `ricc2` program it is recommended to use the `cc2` submenu of the `define` program to set the data groups `$denconv`, `$freeze`, `$cbas`, `$maxcor` and `$rir12`.

Note, that the implementation of non-abelian point groups in `ricc2` is limited to the electronic ground state (but comprises all of the RI-MP2 functionality included in `ricc2`). In the present version `ricc2` can for excited states only deal with abelian point groups (C_1, C_s, C_2, C_i, C_{2h}, C_{2v}, D_2, D_{2h}). The F12 correction can only be calculated in the C_1 point group.

How To Perform a Calculation

Single point calculations:
: Call the `ricc2` program after a converged SCF calculation, which can be carried either with the `dscf` or the `ridft` program.

Geometry optimizations and molecular dynamics:
: Invoke `jobex` with the `-level CC2` option; see Section 5.1 for additional options and parameters of the `jobex` script that might be needed or useful for geometry optimizations and *ab initio* molecular dynamics calculations.

Force constants and vibrational frequencies:

Force constants can be calculated by numerical differentiation of the gradients. Invoke for this `NumForce` with the `-level CC2` option; see Chapter 10 for details about `Numforce`. The usage of the `Numforce` interface for excited states is restricted to C_1 symmetry.

Note: using `ricc2` in connection with `jobex` or `Numforce` requires that the method and the electronic state, for which the gradient should be calculated and written to the interface files, is specified in the option `geoopt` (see Section 9.3.1) in datagroup `$ricc2` (see Section 14.2.13). For calculations on excited states this state has in addition to be included in the input for excitation energies in datagroup `$excitations`.

RI-SCF reference wavefunctions: The `ricc2` can be used in combination with conventional SCF or with the RI-J and RI-JK approximations for SCF, with the exception that the calculation of gradients for a reference wavefunction which employed only the RI-J approximation for the Coulomb matrix but 4-index integrals for the exchange matrix is presently not supported. The implementation gradients in `ricc2` assumes that the reference wavefunction has either been calculated with RI approximation (using `dscf`) or with the RI-JK approximation (using `ridft`).

See Chapter 6 for a discussion of the RI approximations in SCF calculations and 14.2.5 for required input. In geometry optimizations with `jobex` and for the calculation of force constants and vibrational spectra with `NumForce`, the `ricc2` program is used in combination with the RI-JK approximation for the Hatree-Fock calculation (using `ridft`) if `jobex` and `NumForce` are invoked with the `-rijk` option.

How to quote

If results obtained with the `ricc2` program are used in publications, the following citations should be included if you have used the methods, program parts, auxiliary basis sets, or results reported in therein:

Methods:

- for the approximate coupled-cluster singles-and-doubles model CC2:
  O. Christiansen, H. Koch, P. Jørgensen, *Chem. Phys. Lett.*, **243** (1995) 409–418.
- for CI singles with a perturb. correct. for connected double excitations, CIS(D):
  M. Head-Gordon, R. J. Rico, M. Oumi and T. J. Lee, *Chem. Phys. Lett.*, **219** (1994) 21.
  and for the iterative CIS(D_∞) variant:
  M. Head-Gordon, M. Oumi and D. Maurice, *Mol. Phys.* **96** (1999) 593.
- for the algebraic diagrammatic construction through second order ADC(2):
  J. Schirmer, *Phys. Rev. A.* **26** (1981) 2395. A. B. Trofimov and J. Schirmer, *J. Phys. B.* **28** (1995) 2299.

- for the RI-MP2-F12 energy:
  W. Klopper and C. C. M. Samson, *J. Chem. Phys.* **116** (2002) 6397–6410.
  D. P. Tew and W. Klopper, *J. Chem. Phys.* **123** (2005) 074101.
- for the SCS and SOS variants of CC2:
  A. Hellweg, S. Grün, C. Hättig, *Phys. Chem. Chem. Phys.* **10** (2008) 4119-4127.

Implementation:

- please, include always a reference to the publication reporting the implementation of the core part of the `ricc2` program:
  C. Hättig and F. Weigend, *J. Chem. Phys.* **113** (2000) 5154.
- for transition moments and excited state first order properties:
  C. Hättig and A. Köhn, *J. Chem. Phys.* **117** (2002) 6939.
- for triplet excited states include:
  C. Hättig and K. Hald, *Phys. Chem. Chem. Phys.* **4** (2002) 2111.
  C. Hättig, A. Köhn and K. Hald, *J. Chem. Phys.* **116** (2002) 5401.
- for ground geometry optimizations include:
  C. Hättig, *J. Chem. Phys.* **118** (2003) 7751.
- for geometry optimizations for excited states include:
  A. Köhn and C. Hättig, *J. Chem. Phys.* **119** (2003) 5021.
- for calculations with RI-ADC(2), RI-CIS(D), RI-CIS(D_∞) include:
  C. Hättig, *Adv. Quant. Chem.* **50** (2005) 37.
- if the parallel version of `ricc2` is used include a reference to:
  C. Hättig, A. Hellweg, A. Köhn, *Phys. Chem. Chem. Phys.* **8** (2006) 1159.
- for transition moments between excited states:
  M. Pabst and A. Köhn, *J. Chem. Phys.* **129** (2008) 214101.

Appropriate basis sets:

- the appropriate reference for the auxiliary SVP, TZVP and TZVPP basis sets (for calculations with RI-MP2, RI-CC2 and related methods) is:
  F. Weigend, M. Häser, H. Patzelt, R. Ahlrichs, *Chem. Phys. Lett.* **294** (1998) 143.
- for the auxiliary cc-pVXZ (cc-pV(X+d)Z), aug-cc-pVXZ (aug-cc-pV(X+d)Z) basis sets with X = D, T, or Q cite:
  F. Weigend, A. Köhn, C. Hättig, *J. Chem. Phys.* **116** (2001) 3175.
- for the auxiliary cc-pV5Z (cc-pV(5+d)Z), aug-cc-pV5Z (aug-cc-pV(5+d)Z), cc-pwCVXZ with X = D, T, Q, 5 and QZVPP basis sets the reference is:
  C. Hättig, *Phys. Chem. Chem. Phys.* **7** (2005) 59–66.
  This reference should also be included if you employ the analytic basis set gradients implemented in the `ricc2` program for the optimization of your own auxiliary basis set(s).

- for the auxiliary def2-basis sets from Rb to Rn the reference is:
  A. Hellweg, C. Hättig, S. Höfener, and W. Klopper, *Theor. Chem. Acc.* **117** (2007) 587–597.

(For more details on the references for the basis sets included in the basis set libraries of the `TURBOMOLE` distribution see Sec. 1.3 and the library files.)

9.1 CC2 Ground-State Energy Calculations

The CC2 ground-state energy is—similarly to other coupled-cluster energies—obtained from the expression

$$E_{CC} = \langle \mathrm{HF}|H|\mathrm{CC}\rangle = \langle \mathrm{HF}|H\exp(T)|\mathrm{HF}\rangle \quad , \tag{9.1}$$

$$= E_{SCF} + \sum_{iajb} \Big[t_{ij}^{ab} + t_i^a t_j^b\Big]\Big[2(ia|jb) - (ja|ib)\Big], \tag{9.2}$$

where the cluster operator T is expanded as $T = T_1 + T_2$ with

$$T_1 = \sum_{ai} t_{ai}\tau_{ai} \tag{9.3}$$

$$T_2 = \frac{1}{2}\sum_{aibj} t_{aibj}\tau_{aibj} \tag{9.4}$$

(for a closed-shell case; in an open-shell case an additional spin summation has to be included). The cluster amplitudes t_{ai} and t_{aibj} are obtained as solution of the CC2 cluster equations [86]:

$$\Omega_{\mu_1} = \langle \mu_1|\hat{H} + [\hat{H}, T_2]|\mathrm{HF}\rangle = 0 \quad , \tag{9.5}$$

$$\Omega_{\mu_2} = \langle \mu_2|\hat{H} + [F, T_2]|\mathrm{HF}\rangle = 0 \quad , \tag{9.6}$$

with

$$\hat{H} = \exp(-T_1)H\exp(T_1).$$

The residual of the cluster equations $\Omega(t_{ai}, t_{aibj})$ is the so-called vector function. The recommended reference for the CC2 model is ref. [86], the implementation with the resolution-of-the-identity approximation, RI-CC2, was first described in ref. [10].

Advantages of the RI approximation: For RI-CC2 calculations, the operation count and thereby the CPU and the wall time increases—as for RI-MP2 calculations—approximately with $\mathcal{O}(O^2V^2N_x)$, where O is the number of occupied and V the number of virtual orbitals and N_x the dimension of the auxiliary basis set for the resolution of the identity. Since RI-CC2 calculations require the (iterative) solution of the cluster equations (9.5) and (9.6), they are about 10–20 times more expensive than MP2 calculations. The disk space requirements are approximately $O(2V + N)N_x + N_x^2$ double precision words. The details of the algorithms are described in ref. [10], for the error introduced by the RI approximation see refs. [85,89].

Required input data: In addition to the above mentioned prerequisites ground-state energy calculations with the `ricc2` module require only the data group `$ricc2` (see Section 14.2.13), which defines the methods, convergence thresholds and limits for the number of iterations etc. If this data group is not set, the program will carry out a CC2 calculation. With the input

```
$ricc2
  mp2
  cc2
  conv=6
```

the `ricc2` program will calculate the MP2 and CC2 ground-state energies, the latter converged to approximately 10^{-6} a.u. The solution for the single-substitution cluster amplitudes is saved in the file `CCR0--1--1---0`, which can be kept for a later restart.

Ground-State calculations for other methods than CC2: The MP2 equations and the energy are obtained by restricting in the CC2 equations the single-substitution amplitudes t_{ai} to zero. For CCS/CIS the double-substitution amplitudes are excluded from the cluster expansion and in this case the single-substitution amplitudes for the ground state become zero and the energy is identical to the SCF energy. For the Methods CIS(D), CIS(D_∞) and ADC(2) the ground state is identified with the MP2 ground state to define is total energy of the excited state, which is needed for the definition of gradients and (relaxed) first-order properties which are obtained as (analytic) derivatives the total energy.

Fast RI-MP2 calculations with the `ricc2` program: The `ricc2` program includes as a subset also the functionalities of the RI-MP2 program. Because of refined batching algorithms, screening and symmetry treatment the `ricc2` program is usually somewhat faster than the `rimp2` program. This is in particular the cases in the following situations:

- when the molecular point group is D_{2h} or one of its subgroups and a significant number of atoms is positioned on symmetry elements (e.g. planar molecules)
- when because of memory restrictions the `rimp2` program needs many passes of the integral evaluation

All what is needed for a RI-MP2 gradient calculation with the `ricc2` program is:

```
$ricc2
  geoopt model=mp2
```

If you want only the RI-MP2 energy for a single point use as input:

```
$ricc2
  mp2
```

To activate in MP2 energy calculations evaluation of the D_1 diagnostic use:

```
$ricc2
  mp2 d1diag
```

Diagnostics: Together with the MP2 and/or CC2 ground state energy the program evaluates the D_1 diagnostic proposed by Janssen and Nielsen [90], which is defined as:

$$D_1 = \sqrt{\max\left(\lambda_{\max}\Big[\sum_i t_{ai}t_{bi}\Big], \lambda_{\max}\Big[\sum_a t_{ai}t_{aj}\Big]\right)} \tag{9.7}$$

where $\lambda_{\max}[\mathbf{M}]$ is the largest eigenvalue of a positive definite matrix $\mathbf{M}$. (For CC2 the D_1 diagnostic will be computed automatically. For MP2 is must explictly be requested with the `d1diag` option in the `$ricc2` data group, since for RI-MP2 the calculation of D_1 will contribute significantly to the computational costs.) Large values of D_1 indicate a multireference character of the ground-state introduced by strong orbital relaxation effects. In difference to the T_1 and S_2 diagnostics proposed earlier by Lee and coworkers, the D_1 diagnostic is strictly size-intensive and can thus be used also for large systems and to compare results for molecules of different size. MP2 and CC2 results for geometries and vibrational frequencies are, in general, in excellent agreement with those of higher-order correlation methods if, respectively, $D_1(\mathrm{MP2}) \leq 0.015$ and $D_1(\mathrm{CC2}) \leq 0.030$ [90, 13]. For $D_1(\mathrm{MP2}) \leq 0.040$ and $D_1(\mathrm{CC2}) \leq 0.050$ MP2 and/or CC2 usually still perform well, but results should be carefully checked. *Larger values of D_1 indicate that MP2 and CC2 are inadequate to describe the ground state of the system correctly!*

The D_2 diagnostic proposed by Nielsen and Janssen [91] can also be evaluated. This analysis can be triggered, whenever a response property is calculated, e.g. dipole moment, with the keyword `$D2-diagnostic`. *Note that the calculation of D_2 requires an additional $O(N^5)$ step!* $D_2(\mathrm{MP2/CC2}) \leq 0.15$ are in excellent agreement with those of higher-order correlation methods, for $D_2(\mathrm{MP2/CC2}) \geq 0.18$ the results should be carefully checked.

9.2 Calculation of Excitation Energies

With the `ricc2` program excitation energies can at present be calculated with the RI variants of the methods CCS/CIS, CIS(D), CIS(D_∞), ADC(2) and CC2. The CC2 excitation energies are obtained by standard coupled-cluster linear response theory as eigenvalues of the Jacobian, defined as derivative of the vector function with respect to the cluster amplitudes.

$$\mathbf{A}^{\mathrm{CC2}}_{\mu\nu} = \frac{d\Omega_\mu}{dt_\nu} = \begin{pmatrix} \langle\mu_1|[[\hat{H}+[\hat{H},T_2],\tau_{\nu_1}]|\mathrm{HF}\rangle & \langle\mu_1|[\hat{H},\tau_{\nu_2}]|\mathrm{HF}\rangle \\ \langle\mu_2|[\hat{H},\tau_{\nu_1}]|\mathrm{HF}\rangle & \langle\mu_2|[F,\tau_{\nu_2}]|\mathrm{HF}\rangle \end{pmatrix} \tag{9.8}$$

Since the CC2 Jacobian is a non-symmetric matrix, left and right eigenvectors are different and the right (left) eigenvectors E^i_ν ($\bar{E}^i_\mu$) are **not** orthogonal among themselves, but form a biorthonormal basis (if properly normalized):

$$\bar{E}^i E^j = \bar{E}^i_{\mu_1} E^j_{\nu_1} + \bar{E}^i_{\mu_2} E^j_{\nu_2} = \delta_{ij} \quad . \tag{9.9}$$

To obtain excitation energies only the right or the left eigenvalue problem needs to be solved, but for the calculation of transition strengths and first-order properties both, left and right, eigenvectors are needed (see below). A second complication that arises from the non-symmetric eigenvalue problem is that in the case of close degeneracies within the same irreducible representation (symmetry) it can happen that instead of two close lying real roots a degenerate complex conjugated pair of excitation energies and eigenvectors is obtained. CC2 (and also other standard coupled-cluster response methods) are thus not suited for the description of conical intersections etc. For the general theory behind coupled cluster response calculations see e.g. ref. [92, 93] or other reviews.

The `ricc2` program exploits that the doubles/doubles block of the CC2 Jacobian is diagonal and the (linear) eigenvalue problem in the singles and doubles space can be reformulated as a (non-linear) eigenvalue problem in single-substitution space only:

$$\mathbf{A}^{eff}_{\mu_1\nu_1}(t,\omega) = \mathbf{A}^{\mathrm{CC2}}_{\mu_1\nu_1}(t) - \mathbf{A}^{\mathrm{CC2}}_{\mu_1\gamma_2}(t)(\mathbf{A}_{\gamma_2\gamma_2} - \omega)\mathbf{A}^{\mathrm{CC2}}_{\gamma 2\nu_1}(t)$$

$$\mathbf{A}^{eff}_{\mu_1\nu_1}(t^{\mathrm{CC2}},\omega^{\mathrm{CC2}})E_{\nu_1} = \omega^{\mathrm{CC2}} E_{\nu_1}$$

This allows to avoid the storage of the double-substitution part of the eigen- or excitation vectors E_{ν_2}, $\bar{E}_{\nu_2}$. The algorithms are described in refs. [10, 11], about the RI error see ref. [89].

The solution of the CC2 eigenvalue problem can be started from the solutions of the CCS eigenvalue problem (see below) or the trial vectors or solutions of a previous CC2 excitation energy calculation. The operation count per transformed trial vector for one iteration for the CC2 eigenvalue problem is about $1.3 - -1.7$ times the operation count for one iteration for the cluster equations in the ground-state calculation—depending on the number of vectors transformed simultaneously. The disk space requirements are about $O(V + N)N_x$ double precision words per vector in addition to the disk space required for the ground state calculation.

CCS excitation energies are obtained by the same approach, but here double-substitutions are excluded from the expansion of the excitation or eigenvectors and the ground-state amplitudes are zero. Therefore the CCS Jacobian,

$$\mathbf{A}^{\mathrm{CCS}}_{\mu\nu} = \frac{d\Omega_\mu}{dt_\nu} = \langle \mu_1 | [H, \tau_{\nu_1}] | \mathrm{HF} \rangle \quad , \tag{9.10}$$

is a symmetric matrix and left and right eigenvectors are identical and form an orthonormal basis. The configuration interaction singles (CIS) excitation energies are identical to the CCS excitation energies. The operation count for a RI-CIS calculation is $\mathcal{O}(ON^2N_x)$ per iteration and transformed trial vector.

The second-order perturbative correction CIS(D) to the CIS excitation energies is calculated from the expression

$$\omega^{\mathrm{CIS(D)}} = \omega^{\mathrm{CIS}} + \omega^{(D)} = \mathbf{E}^{\mathrm{CIS}}\mathbf{A}^{eff}(t^{\mathrm{MP1}}, \omega^{CIS})\mathbf{E}^{\mathrm{CIS}} \tag{9.11}$$

(Note that t^{MP1} are the first-order double-substitution amplitudes from which also the MP2 ground-state energy is calculated; the first-order single-substitution amplitudes vanish for a Hartree–Fock reference due to the Brillouin theorem.) The operation count for a RI-CIS(D) calculation is similar to that of a single iteration for the CC2 eigenvalue problem. Also disk space requirements are similar.

Running excitation energy calculations: The calculation of excitation energies is initiated by the data group `$excitations` in which at least the symmetries (irreducible representations) and the number of the excited states must be given (for other options see Section 14.2.13). With the following input the `ricc2` program will calculate the lowest two roots (states) for the symmetries A_1 and B_1 of singlet multiplicity* at the CIS, CIS(D) and CC2 level with default convergence thresholds. Ground-state calculations will be carried out for MP2 (needed for the CIS(D) model and used as start guess for CC2) and CC2.

```
$ricc2
  cis
  cis(d)
  cc2
$excitations
  irrep=a1 nexc=2
  irrep=b1 nexc=2
```

The single-substitution parts of the right eigenvectors are stored in files named `CCRE0-`*s*`--`*m*`-`*xxx*, where *s* is the number of the symmetry class (irreducible representation), *m* is the multiplicity, and *xxx* the number of the excitation within the symmetry class. For the left eigenvectors the single-substitution parts are stored in files named `CCLE0-`*s*`--`*m*`-`*xxx*. These files can be kept for later restarts.

Trouble shooting: For the iterative second-order methods CIS(D_∞), ADC(2), and CC2 the solution of the nonlinear partitioned eigenvalue problem proceeds usually in three steps:

1. solution of the CCS/CIS eigenvalue problem to generate reasonable start vectors; the eigenvectors are converged in this step only to a remaining residual norm $<$ `preopt`

*Provided that it is not an unrestricted open shell run. In this case the wavefunctions will not be spin eigenfunctions and multiplicities are not well defined.

2. pre-optimization of the eigenvectors by a robust modified Davidson algorithm (see ref. [10]) using the `LINEAR CC RESPONSE SOLVER` until the norm of all residuals are below `preopt`, combined with a DIIS extrapolation for roots assumed to be converged below the threshold `thrdiis`.

3. solution of the nonlinear eigenvalue problem with a DIIS algorithm using the `DIIS CC RESPONSE SOLVER` until the norm of the residuals are below the required threshold `conv`

This procedure is usually fairly stable and efficient with the default values for the thresholds. But for difficult cases it can be necessary to select tighter thresholds. In case of convergence problems the first thing do is to verify that the ground state is not a multireference case by checking the D1 diagnostic. If this is not the case the following situations can cause problems in the calculation of excitation energies:

- almost degenerate roots in the same symmetry class
- complex roots (break down of the CC approximation close in the neighbourhood of conical intersections)
- large contributions from double excitations

The first two reasons can be identified by running the program with a print level ≤ 3. It will then print in each iteration the actual estimates for the eigenvalues. If some of these are very close or if complex roots appear, you should make sure that the DIIS procedure is not switched on before the residuals of the eigenvectors are small compared to the differences in the eigenvalues. For this, `thrdiis` (controlling the DIIS extrapolation in the linear solver) should be set about one order of magnitude smaller than the smallest difference between two eigenvalues and `preopt` (controlling the switch to the DIIS solver) again about one order of magnitude smaller then `thrdiis`.

Tighter thresholds or difficult situations can make it necessary to increase the limit for the number of iterations `maxiter`.

In rare cases complex roots might persist even with tight convergence thresholds. This can happen for CC2 and CIS(D_∞) close to conical intersections between two states of the same symmetry, where CC response can fail due to its non-symmetric Jacobian. In this case one can try to use instead the ADC(2) model. But the nonlinear partitioned form of the eigenvalue problem used in the `ricc2` program is not well suited to deal with such situations.

Large contributions from double excitations can not be monitored in the output of the (quasi-) linear solver. But it is possible to do in advance a CIS(D) calculation. The CIS(D) results for the `||T2||` diagnostic correlate usually well with the CC2 results for this diagnostic. Else the DIIS solver will print the `||T2||` diagnostics in each iteration if the print level is set > 3. States with large double excitation contributions converge notoriously slow (a consequence of the partitioned formulation used in the `ricc2` program). However, the results obtained with second-order

methods for double excited states will anyway be poor. It is strongly recommended to use in such situations a higher-level method.

9.3 First-Order Properties and Gradients

For the ground state first-order properties (expectation values) are implemented at the SCF, MP2 and CC2 level. Note that for the ground state CCS and CIS are equivalent to SCF. For excited states first-order properties are implemented only at the CCS and CC2 level. Gradients are presently only available for the ground state at the MP2 and the CC2 and for excited states only at the CC2 level.

9.3.1 Ground State Properties, Gradients and Geometries

For CC2, one distinguishes between orbital-relaxed and unrelaxed properties. Both are calculated as first derivatives of the respective energy with respect to an external field corresponding to the calculated property. They differ in the treatment of the SCF orbitals. In the orbital-relaxed case the external field is (formally) already included at the SCF stage and the orbitals are allowed to relax in the external field; in the orbital-unrelaxed case the external field is first applied after the SCF calculation and the orbitals do not respond to the external field. *Orbital-unrelaxed* CC2 properties are calculated as first derivatives of the real part of the unrelaxed Lagrangian [86]

$$\begin{aligned} L^{\text{ur CC2}}(t,\bar{t},\beta) &= \langle \text{HF}|H|\text{CC}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{H} + [\hat{H},T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|\hat{H} + [F_0 + \beta\hat{V},T_2]|\text{HF}\rangle \end{aligned} \tag{9.12}$$

with $H = H_0 + \beta V$—where V is the (one-electron) operator describing the external field, β the field strength, and H_0 and F_0 are the Hamiltonian and Fock operators of the unperturbed system—by the expression:

$$\langle V\rangle^{\text{ur CC2}} = \Re\left(\frac{\partial L^{\text{ur CC2}}(t,\bar{t},\beta)}{\partial\beta}\right)_0 = \sum_{pq} D^{\text{ur}}_{pq}\, V_{pq} \quad , \tag{9.13}$$

$$\begin{aligned} &= \Re\Bigg(\langle \text{HF}|\hat{V}|\text{HF}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{V} + [V,T_2]|\text{HF}\rangle \\ &\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|[\hat{V},T_2]|\text{HF}\rangle\Bigg) \quad , \end{aligned} \tag{9.14}$$

where $\Re$ indicates that the real part is taken. *Relaxed* CC2 properties (and gradients) are calculated from the the full variational density including the contributions from the orbital response to the external perturbation, which are derived from the

Lagrangian [93, 13]

$$
\begin{aligned}
L^{\text{rel CC2}}(t,\bar{t}) &= \langle \mathrm{HF}|H|\mathrm{CC}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{H} + [\hat{H},T_2]|\mathrm{HF}\rangle \qquad (9.15)\\
&\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|\hat{H} + [F,T_2]|\mathrm{HF}\rangle + \sum_{\mu_0} \bar{\kappa}_{\mu_0} F_{\mu_0} \quad ,
\end{aligned}
$$

where F is the Fock operator corresponding to the Hamiltonian of the perturbed system $H = H_0 + \beta V$. One-electron properties are then obtained as:

$$
\begin{aligned}
\langle V\rangle^{\text{rel CC2}} &= \Re\Big(\langle \mathrm{HF}|\hat{V}|\mathrm{HF}\rangle + \sum_{\mu_1} \bar{t}_{\mu_1} \langle \mu_1|\hat{V} + [V,T_2]|\mathrm{HF}\rangle \qquad (9.16)\\
&\quad + \sum_{\mu_2} \bar{t}_{\mu_2} \langle \mu_2|[V,T_2]|\mathrm{HF}\rangle + \sum_{\mu_0} \bar{\kappa}_{\mu_0} V_{\mu_0} \Big) \quad ,\\
&= \sum_{pq} D^{\text{rel}}_{pq}\, V_{pq} \quad . \qquad (9.17)
\end{aligned}
$$

The calculation of one-electron first-order properties requires that in addition to the cluster equations also the linear equations for the Lagrangian multipliers $\bar{t}_\mu$ are solved, which requires similar resources (CPU, disk space, and memory) as the calculation of a single excitation energy. For orbital-relaxed properties also a CPHF-like linear equation for the Lagrangian multipliers $\bar{\kappa}_{\mu_0}$ needs to be solved and the two-electron density has to be build, since it is needed to set up the inhomogeneity (right-hand side). The calculation of relaxed properties is therefore somewhat more expensive—the operation count for solving the so-called Z-vector equations is similar to what is needed for an SCF calculation—and requires also more disk space to keep intermediates for the two-electron density—about $O(2V + 2N)N_x + N_x^2$ in addition to what is needed for the solution of the cluster equations. For ground states, orbital-relaxed first-order properties are standard in the literature.

The calculation of the gradient implies the calculation of the same variational densities as needed for relaxed one-electron properties and the solution of the same equations. The construction of the gradient contributions from the densities and derivative integrals takes about the same CPU time as 3–4 SCF iterations and only minor extra disk space. For details of the implementation of CC2 relaxed first-order properties and gradients and a discussion of applicability and trends of CC2 ground-state equilibrium geometries see ref. [13]. The following is in example input for a MP2 and CC2 single point calculation of first-order properties and gradients:

```
$ricc2
  mp2
  cc2
$response
  static relaxed operators=diplen,qudlen
  gradient
```

A different input is required for geometry optimizations: in this case the model for which the geometry should be optimized must be specified in the data group `$ricc2` by the keyword `geoopt`:

```
$ricc2
  mp2
  cc2
  geoopt model=cc2
```

For CC2 calculations, the single-substitution part of the Lagrangian multipliers $\bar{t}_\mu$ are saved in the file `CCLO--1--1---0` and can be kept for a restart (for MP2 and CCS, the single-substitution part $\bar{t}_\mu$ vanishes).

For MP2 only relaxed first-order properties and gradients are implemented (unrelaxed MP2 properties are defined differently than in CC response theory and are not implemented). For MP2, only the CPHF-like Z-vector equations for $\bar{\kappa}_{\mu_0}$ need to be solved, no equations have to be solved for the Lagrangian multipliers $\bar{t}_\mu$. CPU time and disk space requirements are thus somewhat smaller than for CC2 properties or gradients.

For SCF/CIS/CCS it is recommended to use the modules `grad` and `rdgrad` for the calculation of, ground state gradients and first-order properties.

9.3.2 Excited State Properties, Gradients and Geometries

Also for excited states presently unrelaxed and relaxed first-order properties are available in the `ricc2` program. These are implemented for CCS and CC2. Note, that in the unrelaxed case CIS and CCS are *not* equivalent for excited-states first-order properties and no first-order properties are implemented for CIS in the `ricc2` program.

The unrelaxed first-order properties are calculated from the variational excited states Lagrangian [94], which for the calculation of unrelaxed properties is decomposed into a ground state contribution and Lagrange functional for the excitation energy which leads to expressions for difference densities (or changes of the density matrix upon excitations):

$$L^{\mathrm{ur,ex}}(\bar{N},\bar{E},E,\bar{t},t,\beta) = L^{\mathrm{ur,gs}}(\bar{t},t,\beta) + L^{\mathrm{ur,diff}}(\bar{N},\bar{E},E,\bar{t},t,\beta) \tag{9.18}$$

$$\begin{aligned} L^{\mathrm{ur,diff}}(\bar{N},\bar{E},E,\bar{t},t,\beta) &= \sum_{\mu\nu} \bar{E}_\mu \mathbf{A}_{\mu\nu}(t,\beta) E_\nu \\ &\quad + \sum_{\mu_2} \bar{N}_{\mu_2} \langle \mu_2 | \hat{H} + [F_0 + \beta\hat{V}, T_2] | \mathrm{HF} \rangle \quad , \end{aligned} \tag{9.19}$$

$$\langle V \rangle^{\mathrm{ur,ex}} = \mathcal{R}\left(\frac{\partial L^{ur,ex}(\bar{E},E,\bar{t},t,\beta)}{\partial\beta} \right)_0 \tag{9.20}$$

$$= \sum_{pq} D^{\mathrm{ur,ex}}_{pq} V_{pq} = \sum_{pq} \left(D^{\mathrm{ur}}_{pq} + \Delta D^{\mathrm{ur,ex}}_{pq} \right) V_{pq} \quad , \tag{9.21}$$

with $H = H_0 + \beta V$ and $\Re$ indicating that only the real part is taken, D^{ur} is the unrelaxed ground-state density and $\Delta D_{pq}^{\mathrm{ur,ex}}$ the difference density matrix. The unrelaxed excited-state properties obtained thereby are equivalent to those identified from the second residues of the quadratic response function and are related in the same way to the total energy of the excited states as the unrelaxed ground-state properties to the energy of the ground state. For a detailed description of the theory see refs. [94,93]; the algorithms for the RI-CC2 implementation are described in refs. [89, 12]. ref. [89] also contains a discussion of the basis set effects and the errors introduced by the RI approximation.

In the present implementation, the ground-state and the difference density matrices are evaluated separately. The calculation of excited-state first-order properties thus requires also the calculation of the ground-state density matrix. In addition, the left ($\bar{E}_\mu$) and right (E_μ) eigenvectors and the Lagrangian multipliers $\bar{N}_\mu$ need to be determined for each excited state. The disk space and CPU requirements for solving the equations for $\bar{E}_\mu$ and $\bar{N}_\mu$ are about the same as those for the calculation of the excitation energies. For the construction of the density matrices in addition some files with $\mathcal{O}(n_{root}N^2)$ size are written, where n_{root} is the number of excited states.

The single-substitution parts of the excited-states Lagrangian multipliers $\bar{N}_\mu$ are saved in files named `CCNE0-`*s*`--`*m*`-`*xxx*.

For the calculation of first-order properties for excited states, the keyword `exprop` must be added with appropriate options to the data group `$excitations`; else the input is same as for the calculation of excitation energies:

```
$ricc2
  cc2
$response
  fop unrelaxed_only operators=diplen,qudlen
$excitations
  irrep=a1 nexc=2
  exprop states=all operators=diplen,qudlen
```

Because for calculation of excited-states first-order properties also the (unrelaxed) ground-state density is evaluated, it is recommended to specify also ground-state first-order properties in the input, since they are obtained without extra costs.

To obtain orbital-relaxed first-order properties or analytic derivatives (gradients) the Lagrange functional for the excited state in Eq. (9.18) is—analogously to the treatment of ground states—augmented by the equations for the SCF orbitals and external perturbations are (formally) included in the SCF step, i.e. also in the Fock operator. Since relaxed densities or often computed in connection with geometry optimizations for individual states (rather than simultaneously for many states) a Lagrangian for the total energy of the excited state is used. This has the advantage the only one equation for Lagrangian multipliers for cluster amplitudes ($\bar{N}$) needs to be evaluated instead of two (one for the ground state and one for the energy

difference):

$$L^{\mathrm{rel,ex}}(\bar{E},E,t) = \langle \mathrm{HF}|H|\mathrm{CC}\rangle + \sum_{\mu\nu}\bar{E}_\mu \mathbf{A}_{\mu\nu}(t)E_\nu + \sum_{\mu_2}\bar{N}_{\mu_2}\langle\mu_2|\hat{H}+[F,T_2]|\mathrm{HF}\rangle + \sum_{\mu_0}\bar{\kappa}_{\mu_0}F_{\mu_0} \quad . \tag{9.22}$$

Again the construction of gradients requires the same variational densities as needed for relaxed one-electron properties and the solution of the same equations. The construction of the gradient contributions from one- and two-electron densities and derivative integrals takes approximately the same time as for ground states (approx. 3–4 SCF iterations) and only minor extra disk space. The implementation of the excited state gradients for the RI-CC2 approach is described in detail in Ref. [95]. There also some information about the performance of CC2 for structures and vibrational frequencies of excited states can found.

The following is an example for the CC2 single point calculation for an an excited state gradient (not that in the present implementation it is not possible to compute gradients for several excited states at the same time):

```
$ricc2
  cc2
$excitations
  irrep=a1 nexc=2
  exprop states=all operators=diplen,qudlen
  xgrad states=(a1 2)
```

A different input is again required for geometry optimizations: in this case the model and excited state for which the geometry should be optimized have to be specified in the data group `$ricc2` with the keyword `geoopt`:

```
$ricc2
  geoopt model=cc2 state=(a1 2)
$excitations
  irrep=a1 nexc=2
  exprop states=all operators=diplen,qudlen
```

9.3.3 Visualization of densities and Density analysis

As most other programs which allow for the calculation of wavefunctions and densities also the `ricc2` module is interfaced to wavefunction analysis and visualization toolbox described in chapter 12. From `ricc2` module this interface can used in two different ways

1. If through the `geoopt` keyword in `$ricc2` a unique method and state has been specified for which the density, gradient and properties are evaluated, the density analysis and visualization routines will called by default with the (orbital-relaxed) density for this state and method similar as in `dscf`, `ridft`, `mpgrad`, etc.

2. The `ricc2` program can be called in a special analysis mode which allows to analyse densities and combination (e.g. differences) of densities evaluated in preceeding `ricc2` calculations.

Default density analysis and visualization:
As in a single calculations with the `ricc2` program one–electron densities can be calculated for more than one method and/or electronic state, the interface to the analysis and visualization routines require the specification of a unique level of calculation and a unique state. This is presently done through the `geoopt` flag which determines the method/state for which results are written to interface files (e.g. `control`, `gradient`, or `xxx.map`).

In ground state calculations `ricc2` will pass to the density analysis routines the correlated total (and for UHF based calculations also the spin) density and the canonical SCF orbitals from which the SCF (spin) density is constructed. All options described in chapter 12 are available from within the `ricc2` program apart from the evaluation of electrostatic moments, which would interfere with the calculation of expectation values requested through the `fop` option in `$response`.

In excited state calculation `ricc2` will pass the excited state total (and for UHF based calculation in addition the spin) density. But no ground state densities and/or uncorrelated densities or orbitals. Thus, for excited states the `ricc2` program does, in difference to `egrad` not print out a comparison with the ground state SCF density. Also, all some options which require orbitals (as e.g. the generation and visualization of localized orbitals or some population analysis options) and not available for excited states in `ricc2`.

As other modules, also `ricc2` provides the `-proper` flag to bypass a re-calculation of the density and gradient to enter immediately the density analysis routines with a previously calculated density. The `ricc2` program will then pass the densities found on the interface file for the density analysis routines without further check on the method and state for which they have been evaluated. If both, ground and excited state densities are found on file, both will be passed to the density analysis, thereby providing a shortcut to the `-fanal` and the `$anadens` keyword for the analysis of differences between ground and excited state densities.

The general density analysis option:
In general `ricc2` saves by default all *relaxed* densities generated during a calculation in files named `cc1td-<type>-<mult><irrep>-<number>`, where cc1td stands for "coupled-cluster one-electron total density". `<type>` is one of `mp2-gs` (MP2 ground state), `cc2-gs` (CC2 ground state), `ccs-xs` (CCS excited state), `cc2-xs`

(CC2 excited state), or `adc2-xs` (ADC(2) excited state) and the other entries specify multiplicity, irreducible representation and the number of the state. Having specified the calculation of relaxed densities—e.g. by requesting relaxed one-electron properties or as a by-product of a gradient calculation—you will end up with two files named like

```
cc1td-cc2-gs-1a1-001
cc1td-cc2-xs-3a2-001
```

In case of open shell molecules, additional files with names `cc1sd...` (for one-electron spin-densities) will be generated.

These files are (currently) in a binary format, similar as the files `dens`, `mdens` and `edens`. Therefore be aware that a transfer between different computer architectures may result in trouble.

The densities on these files can be analysed with the tools and interfaces provided by Moloch (see Section 12.2). This can be done by calling `ricc2` with the option `-fanal` which bypasses the usual wavefunction calculation and triggers the program into an analysis mode for densities. In this mode the program interpretes `$anadens` and the keywords described in Section 12.2. To plot, for example, the difference density of the two above mentioned total densities you have to add the following lines in your `control` file

```
$anadens
 calc my_favourite_diffden from
 1d0 cc1td-cc2-xs-3a2-001
-1d0 cc1td-cc2-gs-1a1-001
$pointval
```

and invoke

```
ricc2 -fanal
```

This will generate the files `my_favourite_diffden` and `my_favourite_diffden.map`. The latter can be converted into gOpenMol format as described in Section 12.2.

9.3.4 Fast geometry optimizations with RI-SCF based gradients

If geometry optimizations on MP2 or CC2 level are performed with large basis set, especially with diffuse basis functions, the N^4–steps might become the dominant part of the overall timings. In these cases, the integral screening in the Hartree–Fock part often becomes inefficient. The resolution–of–the–identity can be applied here to speed up the calculation of the HF reference wavefunction, as well as the solution of the coupled–perturbed Hartree–Fock (CPHF) equations in the MP2 or CC2 gradient calculation.

An additional auxiliary basis (denoted `jkbasis`) set has to be assigned via the General Options Menu in the `define` program. In the submenu `rijk` choose `on` and select your auxiliary basis set. Then, run the `jobex` script the additional rijk-flag:

```
> jobex -level cc2 -rijk
```

Note, that it is at the moment not possible to perform these calculation parallel because the RI-JK approximation is presently not supported in the parallel version of the `ridft` program.

9.4 Transition Moments

Transition moments are presently implemented for excitations out of the ground state and for excitations between excited states for the coupled cluster models CCS and CC2. Note, that for transition moments (as excited-state first-order properties) CCS is *not* equivalent to CIS and CIS transition moments are not implemented in the `ricc2` program.

In response theory, transition strengths (and moments) are identified from the first residues of the response functions. Due to the non-variational structure of the coupled cluster models different expressions are obtained for "left" and "right" transitions moments $M^V_{0\leftarrow f}$ and $M^V_{f\leftarrow 0}$ and the transition strengths $S^{0f}_{V_1V_2}$ are obtained as a symmetrized combinations of both:

$$S^{0f}_{V_1V_2} = \frac{1}{2}\left\{M^{V_1}_{0\leftarrow f}M^{V_2}_{f\leftarrow 0} + \left(M^{V_2}_{0\leftarrow f}M^{V_1}_{f\leftarrow 0}\right)^*\right\} \tag{9.23}$$

Note, that only the transition strengths $S^{0f}_{V_1V_2}$ are a well-defined observables but not the transition moments $M^V_{0\leftarrow f}$ and $M^V_{f\leftarrow 0}$. For a review of the theory see refs. [93,96]. The transition strengths calculated by coupled-cluster response theory according to Eq. (9.23) have the same symmetry with respect to interchange of the operators V_1 and V_2 and with respect to complex conjugation as the exact transition moments. In difference to SCF (RPA), (TD)DFT, or FCI, transition strengths calculated by the coupled-cluster response models CCS, CC2, etc. do not become gauge-independent in the limit of a complete basis set, i.e., for example the dipole oscillator strength calculated in the length, velocity or acceleration gauge remain different until also the full coupled-cluster (equivalent to the full CI) limit is reached.

For a description of the implementation in the `ricc2` program see refs. [89, 13]. The calculation of transition moments for excitations out of the ground state resembles the calculation of first-order properties for excited states: In addition to the left and right eigenvectors, a set of transition Lagrangian multipliers $\bar{M}_\mu$ has to be determined and some transition density matrices have to be constructed. Disk space, core memory and CPU time requirements are thus also similar.

The single-substitution parts of the transition Lagrangian multipliers $\bar{N}_\mu$ are saved in files named `CCME0-`*s*`--`*m*`-`*xxx*.

To obtain the transition strengths for excitations out of the ground state the keyword `spectrum` must be added with appropriate options (see Section 14.2.13) to the data group `$excitations`; else the input is same as for the calculation of excitation energies and first-order properties:

```
$ricc2
  cc2
$excitations
  irrep=a1 nexc=2
  spectrum states=all operators=diplen,qudlen
```

To obtain the transition strengths for excitations between excited states the keyword `tmexc` must be added to the data group `$excitations`. Additionally, the initial and final states must be given in the same line; else the input is same as for the calculation of excitation energies and first-order properties:

```
$ricc2
  cc2
$excitations
  irrep=a1 nexc=2
  irrep=a2 nexc=2
  tmexc istates=(a1 1) fstates=all
```

9.5 RI-MP2-F12 Calculations

To obtain the F12 correction to the MP2 energy, the data group `$rir12` must be added to the `control` file. A typical run will include the input:

```
$ricc2
  mp2 energy only
$rir12
```

The MP2-F12 ground-state energy is

$$E_{\text{MP2-F12}} = E_{\text{MP2}} + E_{\text{F12}}, \tag{9.24}$$

where E_{MP2} is the conventional MP2 energy and E_{F12} the correction from explicitly-correlated theory. The second term contains contributions from explicitly-correlated geminal basis functions of the form

$$\hat{Q}_{12} f_{12} |ij\rangle, \tag{9.25}$$

where $|ij\rangle$ is a two-electron determinant of occupied (semi-)canonical Hartree–Fock spin orbitals, f_{12} is a correlation factor, which can be either linear r_{12} (in this case,

the approach is denoted MP2-R12 instead of MP2-F12) or a function of r_{12}, and $\hat{Q}_{12}$ defines the doubles excitation space covered by the geminals (it also ensures strong orthogonality to the occupied orbitals). Usually $\hat{Q}_{12}$ is chosen to be $\hat{Q}_{12} = (1-\hat{O}_1)(1-\hat{O}_2) - \hat{V}_1\hat{V}_2$, where $\hat{O}_\mu = \sum_k |\varphi_k(\mu)\rangle\langle\varphi_k(\mu)|$ is the projection operator onto the space spanned by the occupied spin orbitals φ_k and $\hat{V}_\mu = \sum_a |\varphi_a(\mu)\rangle\langle\varphi_a(\mu)|$ is the projector onto the virtual spin orbitals.

The F12 correction is obtained by minimizing the functional

$$F_{\mathrm{F12}} = \sum_{i<j} \left\{ \mathbf{c}_{ij}^T \mathbf{B}_{ij} \mathbf{c}_{ij} + 2\mathbf{c}_{ij}^T \mathbf{v}_{ij} \right\} \tag{9.26}$$

with respect to the amplitudes collected in the vector $\mathbf{c}_{ij}$. The vectors $\mathbf{v}_{ij}$ and the matrices $\mathbf{B}_{ij}$ are defined as

$$\mathbf{v}_{ij}(kl) = \langle kl | f_{12}\hat{Q}_{12} r_{12}^{-1} | ij \rangle, \tag{9.27}$$

$$\mathbf{B}_{ij}(kl,mn) = \langle kl | f_{12}\hat{Q}_{12}(\hat{f}_1 + \hat{f}_2 - \varepsilon_i - \varepsilon_j)\hat{Q}_{12} f_{12} | mn \rangle, \tag{9.28}$$

in the spin-orbital formalism (m, n denote spin orbitals and $|mn\rangle$ is a two-electron determinant). $\hat{f}_\mu$ is the Fock operator for electron μ and ε_k is a (semi-)canonical Hartree–Fock orbital energy.

A MP2-F12 calculation is defined through a number of choices concerning the nature of the geminals (f_{12} and $\hat{Q}_{12}$), the geminal excitation space (ijkl or ijij) and approximations in computing the B matrix (GBC, EBC, $[\hat{T}, f_{12}]$). These choices correspond to keywords in the `$rir12` data group, explained below.

To run a MP2-F12 calculation, one has to select the auxiliary basis sets `cbas`, `cabs` and optionally `jkbas`. The `ricc2` program uses the robust fitting techniques of Ref. [97] for the F12 integrals and the `cbas` basis is used for both the F12 and the usual MP2 Coulomb integrals. For the density fitting of the Coulomb and exchange matrices of the Fock matrix, the `jkbas` will be used instead of the `cbas` basis if it is included in the control file (this is recommended and is achieved using the `rijk` menu in `define`). For the RI approximation of the 3- and 4-electron integrals as sums of products of 2-electron integrals, intrinsic to the F12 method, the complementary auxiliary basis (CABS) approach is used [98]. If `define` is used to set up the `cabs` basis, the library `cabasen` is searched. This library contains the optimised `cabs` basis sets [99] for the cc-pVXZ-F12 basis sets of Peterson *et al.* [100]. For other basis sets, the auxilliary basis in the library `cabasen` is identical with the auxilliary basis in the library `cbas`.

The `$rir12` data group may be set by choosing the `mp2-f12` option in the `ricc2` menu when running `define`. This command activates the `mp2-f12` menu, where the default options may be changed if desired:

```
INPUT MENU FOR MP2-F12 CALCULATIONS

 ansatz    : CHOOSE ANSATZ                    2      [1,2*,2]
 r12model  : CHOOSE MODEL                     B      [A,A',B]
```

```
comaprox  : COMMUTATOR APPROXIMATION        T+V    [F+K,T+V]
cabs      : CABS ORTHOGONALIZATION          svd  1.0d-08   [cho,svd]
examp     : CHOOSE FORMULATION              fixed  [inv,fixed,noinv]
local     : CHOOSE LOCALIZATION METHOD      off    [off,boys,pipek]
corrfac   : CHOOSE CORRELATION FACTOR       LCG    [R12,LCG]
cabsingles: CABS SINGLE EXCITATIONS         on     [on,off]

* / end    : write $rir12 to file and leave the menu
&          : go back - leaving $rir12 unchanged...
```

`ansatz` corresponds to the choice of $\hat{Q}_{12}$. Almost all modern MP2-F12 calculations use ansatz `2` (default), which gives much improved energies over ansatz `1` (see Ref. [101] for details). The principal additional cost of using ansatz `2` over ansatz `1` is concerned with the coupling between the F12 and conventional amplitudes. This is avoided by choosing `2*`, which corresponds to neglecting EBC (Extended Brillouin Condition) terms in the Fock matrix elements.

`r12model` is the method of computing the matrices $\mathbf{B}_{ij}$ (see Ref. [101] for details). The cost and accuracy increases in the progression `A`, `A'`, `B`. It is recommended to use `B` (default). The energies computed using `A` are then also printed out in the output.

`comaprox` is the method for approximately computing the integrals for the operator $[\hat{T}, f_{12}]$, where the matrix representations of `F+K` or `T+V` are used. `T+V` (the core Hamiltonian) is recommended and is the default.

`cabs` refers to the method of orthogonalising the orbitals in the complementary auxiliary basis. Single-value decomposition (`svd`) or Choleski decomposition (`cho`) are available. `svd` is recommended and is the default, with a threshold of `1.0d-08`. The basis set used for CABS is set from the `ricc2` menu.

`examp` refers to the choice of excitation space. `inv` is the orbital-invariant merhod of Ref. [102], with amplitudes $c_{ij}(kl)$. `noinv` is the original orbital-dependent diagonal "ijij" method of Ref. [102], with amplitudes $c_{ij}(ij)$ (not recommended, unless in combination with localised orbitals). `fixed` is the (diagonal and orbital-invariant) rational generator approach of Ref. [103], where the F12 amplitudes are not optimised, but predetermined using the coalescence conditions (default).

`local` controls which orbitals are used in the calculation. `off` means that (semi-)canonical Hartree–Fock orbitals are used (default). For calculations using linear-r_{12} as correlation factor, and r12model `A` or

A', localised orbitals may be used. Both the Boys [104] and Pipek–Mezey [105] methods are available for localisation of the orbitals.

`corrfac` corresponds to the choice of correlation factor f_{12} in the geminal basis functions. `R12` results in a calculation using linear-r_{12} and `LCG` results in a calculation using the Slater-type correlation factor with exponent 1.4 a_0^{-1}, represented as a linear combination of six Gaussians (see Ref. [106]). Note that the exponents 0.9, 1.0 and 1.1 a_0^{-1} are recommended for use with the cc-pVXZ-F12 basis sets [100].

`cabsingles` switches on/off the calculation of a second-order correction to the Hartree–Fock energy by accounting for single excitations into the complementary auxiliary basis set (CABS). The single excitations into the CABS basis can be computed without extra costs if the CABS Fock matrix elements are required anyway for the F12 calculation (*i.e.*, for ansatz `2`, approximation `B` or comaprox `F+K`). The computation of CABS singles cannot be switched off if it is free of costs.

Further options:

`corrfac LCG` refers to a further data group for the definition of the correlation factor. When `define` is used, the default is

```
$lcg
  nlcg    6
  slater  1.4000
```

The nature of the LCG correlation factor may be changed by editing this data group in the control file. For example, to use a Slater-type correlation factor with exponent 1.0 a_0^{-1}, represented as a linear combination of three Gaussians, use

```
$lcg
  nlcg    3
  slater  1.0000
```

Alternatively, the exponents and coefficients of the fit may be given explicitly:

```
$lcg
  nlcg    3
  expo1  coef1
  expo2  coef2
  expo3  coef3
```

`pairenergy` controls whether or not the F12 contribution to the MP2 pair energies appear in the output (default `off`),

```
  pairenergy  off      [on,off]
```

MP2-F12 calculations may be combined with Grimme's SCS approach (S. Grimme, *J. Chem. Phys.* **118** (2003) 9095.) by inserting `scs` in `$ricc2`,

```
$ricc2
  mp2 energy only
  scs
```

In this case, the SCS parameters cos=6/5 and css=1/3 are used. Also individual scaling factors for the same-spin and opposite-spin contributions may be defined, see Section 9.7.

For open-shell calculations, two choices of the `examp fixed` method are available. These are controled by a keyword in the `$rir12` data group

```
  ump2fixed  full      [diag,full]
```

These differ in the treatment of the $\alpha\beta$ block, where either only the diagonal excitations enter (with amplitude 0.5) `diag`, or the equivalent of the spin-adapted singlet and triplet pair excitations enter (as far as possible) `full`. Note that the `diag` method with UMP2-F12 yields a result different to that of `fixed` MP2-F12, even for identical RHF and UHF determinants. However, the `diag` method is somewhat less expensive than the `full` method.

Recommendations for orbital and auxiliary basis sets:

The best orbital basis sets to use for MP2-F12 calculations are probably the cc-pVXZ-F12 basis sets, specially optimised for MP2-F12 calculations [100] for the atoms H, He, B–Ne and Al–Ar. In conjunction with these cc-pVXZ-F12 basis sets, we recommend to use the optimised cc-pVXZ-F12 sets of Yousaf and Peterson [99] as `cabs`. Furthermore, `cbas` and `jkbas` basis sets can be selected from the `cbasen` and `jkbasen` libraries, respectively, using the alias cc-pVXZ-F12 (a `jkbas` is currently not available for He, Ne and Ar). This alias points to the corresponding aug-cc-pwCV(X+1)Z `cbas` and aug-cc-pV(X+1)Z `jkbas`. These recommendations are on the side of caution and are likely to be refined as more experience is gained [88, 107, 108].

For atoms other than H, He, B–Ne and Al–Ar, optimised F12 basis sets are not yet available. In this case, basis sets must be selected and/or optimised carefully. It is advised to contact the Theoretical Chemistry Group in Karlsruhe for support (e-mail to: `klopper@chem-bio.uni-karlsruhe.de`).

9.6 Parallel RI-MP2 and RI-CC2 Calculations

The `ricc2` program is partially parallized for distributed memory architectures (e.g. clusters of Linux boxes) based on the *message passing interface* (MPI) standard. In the present version parallel calculations can be carried out for ground state and excitation energies for all wavefunction models available in `ricc2`. The analytic gradients for RI-MP2 and RI-CC2 in the ground state and RI-CC2 in excited states are also parallized.

While in general the parallel execution of `ricc2` works similar to that of other parallized Turbomole modules (as e.g. `dscf` and `grad`), there are some important difference concerning in particular the handling of the large scratch files needed for RI-CC2 (or RI-MP2). As the parallel version `dscf` also the parallel version of `ricc2` assumes that the program is started in a directory which is readable (and writable) on all compute nodes under the same path (e.g. a NFS directory). The directory must contain all input files and will at the end of a calculation contain all output files. Large scratch files (e.g. for integral intermediates) will be placed under the path specified in the `control` file with `$tmpdir` (see Section 14.2.13) which should point to a directory in a file system with a good performance. The parallel version of the `ricc2` program can presently account for the following two situations:

Clusters with single processor nodes and local disks: Specify in `$tmpdir` a directory in the file system on the local disk. All large files will be places on the nodes in these file systems. (The local file system must have the same name on all nodes)

Clusters with multiple (e.g. dual) processor nodes and local disks Set in addition to `$tmpdir` the keyword `$sharedtmpdir` to indicate that several processes might share the same local disk. The program will than create in s the directory given in `$tmpdir` subdirectories with node-specific names.

Note that at the end of a `ricc2` run the scratch directories specified with `$tmpdir` are not guaranteed to be empty. To avoid that they will fill your file system you should remove them after the `ricc2` calculation is finished.

Another difference to the parallel HF and DFT (gradient) programs is that `ricc2` will communicate much larger amounts of data between the compute nodes. With a fast network interconnection (Gigabit or better) this should not cause any problems, but with slow networks the communication might become the limiting factor for performance or overloading the system. If this happens the program can be put into an alternative mode where the communication of integral intermediates is replaced by a reevaluation of the intermediates (at the expense of a larger operation count) wherever this is feasible. Add for this in the `control` the following data group:

```
$mpi_param
  min_comm
```

9.7 Spin-component scaling approaches (SCS/SOS)

By introducing individually scaling factors to the same–spin and opposite–spin contributions of the correlation energy second–order methods can be modified for a (hopefully) better performance. SCS-MP2 has first been proposed by S. Grimme and SOS-MP2 by Y. Jung *et al.* (see below). The generalization of SCS and SOS to CC2 for ground and excited states is described in [109]. It uses the same scaling factors as proposed for the original SCS- and SOS-MP2 approaches (see below). In the `ricc2` program we have also implemented SCS and SOS variants of CIS(D) for excitation energies and of CIS(D_∞) and ADC(2) for excitation energies and gradients, which are derived from SCS-CC2 and SOS-CC2 in exactly the same manner as the unmodified methods can be derived as approximations to CC2 (see Sec. 9.2 and ref. [110]). Please note, that the SCS-CIS(D) and SOS-CIS(D) approximations obtained in this way and implemented in `ricc2` differ from the spin-component scaled SCS- and SOS-CIS(D) methods proposed by, respectively, S. Grimme and E. I. Ugorodina in [111] and Y. M. Rhee and M. Head–Gordon in [112].

A line with scaling factors has to be added in the `$ricc2` data group:

```
$ricc2
scs  cos=1.2d0  css=0.3333d0
```

`cos` denotes the scaling factor for the opposite–spin component, `css` the same–spin component.

As an abbreviation
`scs`
can be inserted in `$ricc2`. In this case, the SCS parameters cos=6/5 and css=1/3 proposed S. Grimme (S. Grimme, *J. Chem. Phys.* **118** (2003) 9095.) are used. These parameters are also recommended in [109] for the SCS variants of CC2, CIS(D), CIS(D_∞), and ADC(2) for ground and excited states.

Also, just
`sos`
can be used as a keyword, to switch to the SOS-MP2 approach proposed by the Head-Gordon group with scaling factors of cos=1.3 and css=0.0 (Y., Jung, R.C. Lochan, A.D. Dutoi, and M. Head-Gordon, *J. Chem. Phys.* **121** (2004) 9793.), which are also recommended for SOS variants of CC2, CIS(D), CIS(D_∞), and ADC(2).

Restrictions:

- the spin (S^2) expectation value for open-shell calculation can not be evaluated in the SCS or SOS approaches

9.7.1 Laplace-transformed SOS-RI-MP2 with $\mathcal{O}(\mathcal{N}^4)$ scaling costs

The `ricc2` module contains now also a first implementation of SOS-MP2 which exploits the RI approximation and a Laplace transformation of the orbital energy denominators

$$\frac{1}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} = \int_0^\infty e^{-(\epsilon_a+\epsilon_b-\epsilon_i-\epsilon_j)t} dt \approx \sum_{\alpha=1}^{n_L} w_\alpha e^{-(\epsilon_a+\epsilon_b-\epsilon_i-\epsilon_j)t_\alpha} \quad , \tag{9.29}$$

to achieve an implementation with $\mathcal{O}(\mathcal{N}^4)$ scaling costs, opposed to the conventional $\mathcal{O}(\mathcal{N}^5)$ scaling implementation. In particular for large molecules the Laplace-transformed implementation can reduce a lot the computational costs of SOS-MP2 calculations without loss in accuracy.

The Laplace-transformed implementation for SOS-MP2 calculations is activated with the input

```
$laplace
   conv=5
```

where the parameter `conv` is a convergence threshold for the numerical integration in Eq. (9.29). A value of `conv=5` means that the numerical integration will be converged to a root mean squared error of $\approx 10^{-5}$ a.u.

Whether the conventional or the Laplace-transformed implementation will be more efficient depends firstly on the system size (the number of occupied orbitals) and secondly on the required accuracy (the number of grid points for the numerical integration in Eq. (9.29)) and can be understood and estimated from the following considerations:

- The computational costs for the most expensive step in (canonical) RI-MP2 energy calculations for large molecules requires $\frac{1}{2}O^2V^2N_x$ floating point multiplications, where O and V are, respectively, the number occupied and virtual orbitals and N_x is the number of auxiliary functions for the RI approximation. For the LT-SOS-RI-MP2 implementation the most expensive step involves $n_L OVN_x^2$ floating point multiplications, where n_L is the number of grid points for the numerical integration. Thus, the ratio of the computational costs is approximately

  $$conv : LT \approx \frac{\frac{1}{2}O^2V^2N_x}{n_L OVN_x^2} = \frac{OV}{2n_L N_x} \approx O : 6n_L \quad ,$$

  where for the last step $N_x \approx 3V$ has been assumed. Thus, the Laplace-transformed implementation will be faster than the conventional implementation if $O > 6n_L$.

The number of grid points n_L depends on the requested accuracy and the spread of the orbital energy denominators in Eq. (9.29). The efficiency of Laplace-transformed

SOS-RI-MP2 calculations can therefore (in difference to conventional RI-MP2 calculations) be enhanced significantly by a carefull choice of the thresholds, the basis set, and the orbitals included in the correlation treatment:

- The threshold `conv` for the numerical integration is by default set to the value of `conv` specified for the ground state energy in the data group `$ricc2` (see Sec. 14.2.13), which is initialized using the threshold `$denconv`, which by default is set conservatively to the tight value of 10^{-7}.
  - For single point energy calculations `conv` in `$laplace` can savely be set to 4, which gives SOS-MP2 energies converged within $\approx 10^{-4}$ a.u. with computational costs reduced by one third or more compared to calculations with the default settings for these thresholds.
  - For geometry optimizations with SOS-MP2 we recommend to set `conv` in `$laplace` to 5.
- The spread of the orbital energy denominators depends on the basis sets and the orbitals included in the correlation treatment. Most segmented contracted basis sets of triple-ζ or higher accuracy (as e.g. the TZVPP and QZVPP basis sets) lead to rather high lying "anti core" orbitals with orbital energies of 10 a.u. and more.
  - For the calculation of SOS-MP2 valence correlation energies it is recommended to exclude such orbitals from the correlation treatment (see input for `$freeze` in Sec. 14).
  - Alternatively one can use general contracted basis sets, as e.g. the correlation consistent cc-pVXZ basis sets. But note that general contracted basis sets increase the computational costs for the integral evaluation in the Hartree-Fock and, for gradient calculations, also the CPHF equations and related 4-index integral derivatives.
  - Also for the calculation of all-electron correlation energies with core-valence basis sets which include uncontracted steep functions it is recommended to check if extremely high-lying anti core orbitals can be excluded.

Note that for large molecules it is recommended to disable for geometry optimizations (or for gradient or property calculations in general) the preoptimization for the Z vector equations with the `nozpreopt` option in the `$response` data group (see Sec. 14.2.13).

Restrictions:

- The Laplace-transformed SOS-MP2 implementation is presently only parallelized with MPI. The OpenMP parallelization is not (yet) recognized by the LT-SOS-RI-MP2 related program parts.

- It is presently not compatible with the calculation of the D_1 and D_2 diagnostics. The respective options will be ignored by program if the Laplace-transformed implementation is used.

Chapter 10

Calculation of Vibrational Frequencies and Vibrational Spectra

Calculation of second derivatives of total energies leads to the molecular Hessian, which enables prediction of vibrational frequencies and infrared spectra (within the harmonic approximation) as well as the application of improved algorithms for geometry optimization and transition state search.

The `aoforce` module calculates analytically harmonic vibrational frequencies within the HF- or (RI)DFT-methods for closed-shell- and spin-unrestricted open-shell-systems. Broken occupation numbers would lead to results without any physical meaning. Note, that RI is only used partially, which means that the resulting Hessian is only a (very good) approximation to exact second derivatives of the RIDFT-energy expression. Apart from a standard force constant calculation which predicts all (symmetry allowed and forbidden) vibrational transitions, it is also possible to specify certain irreps for which the calculation has to be done exclusively or to select only a small number of lowest eigenvalues (and eigenvectors) that are generated at reduced computational cost.

Furthermore, the `Numforce` script allows the calculation of second derivatives for all methods for which a program for analytic gradients is available in `TURBOMOLE`, i.e. the main use of this script is the prediction of vibrational spectra at the MP2 level and for excited states using RI-CC2 or TDDFT.

If force constant calculations result in imaginary frequencies, molecular distortions along these normal modes should lower the energy. To distort the molecule, use the interactive module `vibration`, output of the new coordinates is done to the general input file on `$newcoord`.

Vibrational frequencies also enable calculation of the molecular partition function and thus prediction of thermodynamic functions at temperatures other than 0 K

and finite pressure (within the assumption of an ideal gas and no coupling between degrees of freedom). These functions can be obtained with the interactive module `Freeh`, results are printed to standard I/O.

Prerequisites

1. Both `aoforce` and even more `Numforce` require well converged SCF-/DFT-calculations (e.g. `$scfconv 8` and `jobex [-ri] -gcart 4`).
2. The maximum core memory the program `aoforce` is allowed to allocate should be defined in the data group `$maxcor`; the recommended value is about 50% of the available (physical) core memory (in case of RI-calculations subtract the memory specified in `$ricore`).
3. To start `aoforce` in the lowest eigenvalue search mode, use the keyword `$les`. For its use as well as other keywords dealing with the calculation of only some irreps, see the Referenceguide part of this manual.
4. `Numforce` additionally requires the file `gradient` and will not work, if the calculation is not done at a stationary point of the molecular total energy. For reliable results, always use `Numforce` with the option `-central` (i.e. central differences) and be aware of effects due to the step length (option `-d` *real*;, default value is 0.02 a.u.). It is strongly recommended to use `Numforce` in DFT calculations only with the option `weight derivatives` in `$dft`, since this provides more accurate gradients and thus frequencies, see Section 14.2.8.
5. The `Numforce` script can be run for different levels of theory, which means that the binaries it calls have to be specified additionally. To perform calculations using the RI approximation, call `Numforce` with the option `-ri`. MP2 and CC2 calculations are requested via the options `-level mp2` and `-level cc2`, respectively. To select the correct option(s), use the explanations you get by calling `NumForce -h`.

For a review of theory and implementation see refs. [113, 114].

Limitations

The `aoforce` code has presently a number of limitations one should be aware of:

- It can only handle basis sets up to at most g functions.
- Point groups with reducible E-representations (such as C_n and C_{nh} with $n \geq 3$, S_n with $n \geq 5$, or T and T_d)
- Frozen internal or cartesian coordinates are not recognized. `aoforce` will allways evaluate the full hessian matrix.

10.1 Analysis of Normal Modes in Terms of Internal Coordinates

A note in advance: The analysis of normal modes can (at nearly no computational cost) always be redone as long as you keep a copy of the file `hessian`.

A general prerequisite for this option is that you have defined a set of non-redundant coordinates for all 3N-6 (3N-5) degrees of freedom of your molecule. To make sure that this is the case, you should switch off redundant coordinates (currently, this is only possible by manually removing the data group `$redundant` and also removing the entry `redundant on` in `$optimize`). Run `define` to generate non-redundant coordinates by using the `iaut` command in the internal coordinate menu (or by creating them manually via `idef`). We recommend to use the `irem` command first to delete all previous definitions of internal coordinates. See Section 4 for further details. If the molecules point group is not C_1, `define` will set some of the coordinate to status `d` (display) or `i` (ignore). Use the `ic` command to change all coordinates to `k`. You can also achieve this by editing in the `$intdef` data-group manually.

The analysis in internal coordinates is switched on by adding a line in the data-group `$drvopt` that has the following syntax:

`analysis [only] intcoord [print` *print-level*`]`

Keywords in square brackets are optional. If `only` is added, the program assumes that the file `hessian` exists and runs only the analysis part of `aoforce`. The program will give the following output (controlled by the print level given in parenthesis):

- diagonal elements of the Hessian in internal coordinates (force constants of bonds, angles, etc.) (print level 0)
- complete force constant matrix in internal coordinates (print level 2)
- normal modes in terms of internal coordinates (print level 1)
- Potential energy contributions $\tilde{V}_{ij}^n$, defined as

  $$\tilde{V}_{ij}^n = L_i^n L_j^n F_{ij}/\omega^n$$

  where L_i^n are the elements of the normal coordinate belonging to mode n and F_{ij} are the elements of the force constant matrix, both expressed in the internal coordinate basis; ω is the related eigenvalue. The program will list the diagonal contributions $\tilde{V}_{ii}^n$ (print level 1), the off-diagonal contributions $\tilde{V}_{ij}^n + \tilde{V}_{ji}^n = 2\tilde{V}_{ij}^n$ (print level 2 for up to 10 atoms, else print level 10) and the brutto contributions $\sum_i \tilde{V}_{ij}^n$ (print level 1).
- Based on these quantities, the program will give an assignment of normal modes by listing all internal coordinates with large diagonal or brutto contributions (print level 0).

Note that for large molecules or complicated topologies the B-matrix (that is used to transform from Cartesian coordinates into internal coordinates and vice versa) may become singular. In this case only the normal modes in the internal coordinate basis can be listed.

10.2 Calculation of Raman Spectra

Vibrational Raman scattering cross sections are computed in the approximation of the polarizability theory from derivatives of the frequency-dependent polarizability tensor with respect to normal modes of vibration,

$$\left(\frac{d\sigma}{d\Omega}\right) = k_\omega \left(c_i \alpha'^2(\omega) + c_a \gamma'^2(\omega)\right) .$$

Here, $\alpha'^2(\omega)$ and $\gamma'^2(\omega)$ denote the isotropic part and the anisotropy of the differentiated polarizability tensor, respectively. The coefficients c_i and c_a depend on the scattering geometry and the polarization of the incident and scattered radiation. The factor

$$k_\omega = \frac{\hbar}{4\pi\epsilon_0^2 c^4} \frac{(\omega - \omega_v)^4 g_v}{2\omega_v}$$

includes the frequency ω_v and the degeneracy g_v of the vibration. c is speed of light and ϵ_0 stands for the dielectric constant of vacuum.

Computation of Raman spectra with `TURBOMOLE` is a three-step procedure. First, vibrational frequencies and normal modes are calculated by `aoforce`. Cartesian polarizability derivatives are computed in the second step by `egrad`, see Section 7.4.7. Finally, the program `intense` is used to project the polarizability derivatives onto vibrational normal modes and to compute Raman scattering cross sections which are written out along with vibrational frequencies and normal modes. The script `Raman` can be used to perform all these steps automatically.

10.3 Vibrational frequencies with fixed atoms using `NumForce`

The `NumForce` script provides with the option `-frznuclei` a possibility to do a vibrational analysis with fixed atoms. The atoms for which the cartesian coordinates should frozen have to be marked in `$coord` with a "f" behind the atom type. The frozen coordinates will be skipped during the numerical evaluation of the force constant matrix; instead all off-diagonal elements of the force constant matrix which refer to one or two frozen coordinates will be set to zero, while the diagonale elements for the frozen coordinates will be set to an arbitrarly chosen large value.

This feature is mainly intended to allow for a vibrational analysis in embbeded cluster calculations e.g. for defects in ionic crystals. The vibrational analysis uses a kind of "frozen phonon" approximation which corresponds to setting the masses of the fixed atoms to infinity, i.e. decoupling the fixed atoms mechanically from the "mechanically active" subsystem. The resulting vibrational frequencies will thus only

provide good approximations to the true (harmonic) frequencies for such modes for which the mechanical coupling to the embedding environment is negligible. In particular the frequencies of stretch modes which involve bonds between the "mechanically active" subsystem and atoms with frozen coordinates will be strongly affected by this approximation.

Note:

- The `-frznuclei` is not compatible with the polyhedral difference algorithm. It can only be used with central differences which should be enforced with the `-central` option.
- If the option `-frznuclei` is switched on, the program assumes that the constraints enforced by fixing coordinates remove the six external degrees of freedom for on overall rotation or translation of the system and therefore the hessian matrix is not projected onto the subspace of internal coordinates. Fixing the coordinates of only one or two atoms might does lead to some artifical small, but non-zero frequencies.
- Zero-point vibrational energies calculated with the `-frznuclei` option are only meaningful for comparison of systems with the same mechanically active atoms and similar embedding, as the contributions from the frozen coordinates are not included.

Chapter 11

Calculation of NMR Shieldings

The program `mpshift` calculates nuclear magnetic shielding constants using the GIAO (Gauge Including Atomic Orbital) method.

At present the following methods are implemented:

HF-SCF	the coupled perturbed Hartree–Fock (CPHF) equations in the AO basis are solved using a semi-direct iterative algorithm [115] similar to `dscf`.
DFT	using either non-hybrid functionals where no iterations are needed [116] or hybrid functionals where the same algorithm as at the HF-SCF level is used.
MP2	semi-direct method, see ref. [17].

11.1 Prerequisites

1. `mpshift` needs converged MO vectors from a SCF or DFT run (`dscf` or `ridft`)
2. for SCF or DFT calculations, no specifications have to be made in the `control` file
3. it is not possible to run the program in the fully direct mode when doing an SCF, MP2 or a DFT (using hybrid functionals) run, so you will have to perform a statistics run of `dscf` before calling `mpshift`, or just set the size of the `$twoint` file to a non-zero value
4. to perform an MP2 calculation of the NMR shieldings you have to prepare the input with `mp2prep -c`

11.2 How to Perform a SCF of DFT Calculation

All you have to do for running `mpshift` is typing `mpshift` at the shell level.

The results of a SCF or DFT calculation (the trace of the total shielding tensors, its anisotropy and the CPHF contribution for each symmetry distinct atom) are written into the `control` file after the keyword `$nmr <rhf/dft> shielding constants`.

This data group is write only for `mpshift`, but you can utilize it for graphical rendering of the calculated NMR spectra and for a quick overview of the results. A more detailed output with the complete shielding tensors can be found in the output of `mpshift`, so it is recommended to put the output in a file when calling the program.

11.3 How to Perform a MP2 calculation

To perform an MP2 calculation of the NMR shieldings you have to prepare the input with `mp2prep -c`.

`mpshift` will then calculate both the SCF and MP2 shielding constants. The result is written into the `control` file after the keyword `$nmr mp2 shielding constants`.

The script `mp2prep` will create the keywords

```
$csmp2
$thize      .10000000E+10
$mointunit
 type=intermed unit=61 size=0 file=halfint
 type=1112     unit=63 size=0 file=moint#1
 type=1122     unit=64 size=0 file=moint#j
 type=1212     unit=65 size=0 file=moint#k
 type=1212a    unit=70 size=0 file=moint#a
 type=gamma#1  unit=71 size=0 file=gamma#1
 type=gamma#2  unit=72 size=0 file=gamma#2
 type=dtdb#1   unit=76 size=0 file=dtdb#1
 type=dtdb#2   unit=77 size=0 file=dtdb#2
$traloop 1
$statistics mpshift
```

and starts a statistics run of mpshift (by calling `mpshift`). If the resulting disk space requirement exceeds the automatically detected free disk space on your system, it will increase `$traloop` and run a statistics run again. This will be done as long as your free disk space is not sufficient for the calculation.

If the `mp2prep` script fails to run on your system, try to use the -p option or do the procedure described above by hand. Call `mp2prep -h` for more informations about `mp2prep`.

11.4 Chemical Shifts

NMR shifts are obtained by comparing nuclear shieldings of your test compound with a reference molecule ($\delta_{subst} = \delta_{ref} + \sigma_{ref} - \sigma_{subst}$). Therefore you have to choose a reference molecule with a well-known shift for which you can easily calculate the absolute shielding constant. This implies a certainty about the geometry, too. Furthermore you have to use the very same basis set for corresponding atoms to minimize the basis set influence.

Keywords for the module Mpshift

A list of keyword for the module `mpshift` can be found in Section 14.2.19.

11.5 Other Features and Known Limitations

- the `mpshift` program can be restarted at any stage of computing, since all intermediate results are written into the file `restartcs`. In case of an external program abort you have to remove the `$actual step` flag (by the command `actual -r` or using an editor). `mpshift` analyses this file and decides where to continue
- ECPs can not be used since the electrons in the ECP cores are not taken into account
- molecular point groups that contain reducible e representations are not supported (C_n, C_{nh} with $n > 2$)
- as in `mpgrad`, basis sets with a contraction that is greater than 10 are currently not supported
- PBE and PBE0 DFT functionals are not implemented in `mpshift`

Chapter 12

Molecular Properties, Wavefunction Analysis, and Interfaces to Visualization Tools

12.1 Wavefunction analysis and Molecular Properties

Molecular properties (electrostatic moments, relativistic corrections, population analyses for densities and MOs, construction of localized MOs, etc.) can be calculated with the module `moloch`. Note that this program does not support unrestricted open-shell input (a script called `moloch2` can currently be used as a work-around; type `moloch2 -help` for further information). Moreover, analyses of densities apart from those calculated from molecular orbitals (e.g. MP2 densities, densities of excited states) are not possible. For the current version of moloch we refer to the keywords listed in Section 14.2.16 which partly can also be set by `define` (see also Chapter 4).

Note: `moloch` is no longer supported, but

most functionalities of `moloch` now are integrated in programs that generate MOs or densities and can be done directly within the modules `dscf`, `ridft`, `rimp2`, `mpgrad`, `ricc2` and `egrad`. If (some of) following keywords are set, corresponding operations will be performed in the end of these programs. If one desires to skip the MO- or density generating step, in case of programs `dscf`, `ridft`, `rimp2` and `mpgrad` it is possible to directly jump to the routine performing analyses by typing "`<program> -proper`". Currently, the respective keywords have to be inserted in the `control` file by hand (not by `define`).

Here we briefly present the functionalities (i.e. the default use of keywords), non-default suboptions are described in detail in Section 14.2.17.

Electrostatic moments: up to quadrupole moments are calculated by default for the above modules.

Relativistic corrections: `$mvd` leads to calculation of relativistic corrections for the SCF total density in case of `dscf` and `ridft`, for the SCF+MP2 density in case of `rimp2` and `mpgrad` and for that of the calculated excited state in case of `egrad`. Quantities calculated are expectation values $< p^2 >, < p^4 >$ and the Darwin term ($\sum 1/Z_A * \rho(R_A)$). Note, that at least the Darwin term requires an accurate description of the cusp in the wave function, thus the use of basis sets with uncontracted steep basis functions is recommended. Moreover note, that when using of ECPs these quantities are not too reasonable (a respective warning is written to the output).

Population analyses: Population analyses are driven by the keyword `$pop`.

Without any extension Mulliken population analyses (MPA) are carried out for all densities present in the respective program, e.g. total (and spin) densities leading to Mulliken charges (and unpaired electrons) per atom in RHF(UHF)-type calculations in `dscf` or `ridft`, SCF+MP2 densities in `rimp2` or `mpgrad`, excited state densities in `egrad`. Suboptions (see Section 14.2.17) also allow for calculation of Mulliken contributions of selectable atoms to selectable MOs including provision of data for graphical output (simulated density of states).

With `$pop nbo` a Natural Population Analysis (NPA) [117] is done. Currently only the resulting charges are calculated.

With `$pop paboon` a population analyses based on occupation numbers [118] is performed yielding "shared electron numbers (SENs)" and multicenter contributions. For this method always the total density is used, i.e. the sum of alpha and beta densities in case of UHF, the SCF+MP2-density in case of MP2 and the GHF total density for (two-component-)GHF. Note that the results of such an analysis may depend on the choice of the number of modified atomic orbitals ("MAOs"). By default, numbers of MAOs which are reasonable in most cases are taken (see Section 14.2.17). Nevertheless it is warmly recommended to carefully read the information concerning MAOs given in the output before looking at the numbers for atomic charges and shared electron numbers. For different ways of selecting MAOs see Section 14.2.17.

Generation of localized MOs: `$localize` enables calculation of localized molecular orbitals. Per default a Boys localization including all occupied MOs is carried out (i.e. the squared distance of charge centers of different LMOs is maximized). As output one gets localized MOs (written to files `lmos` or `lalp` /`lbet` in UHF cases), informations about dominant contributions of canonical MOs to LMOs and about location of LMOs (from Mulliken PA) are written to standard output.

Fit of charges due to the electrostatic potential: `$esp_fit` fits point charges at the positions of nuclei to electrostatic potential arising from electric charge dis-

tribution (for UHF cases also for spin density, also possible in combination with `$soghf`). For this purpose the ("real") electrostatic potential is calculated at spherical shells of grid points around the atoms. By default, Bragg-Slater radii, r_{BS}, are taken as shell radii.

A parametrization very close to that suggested by Kollman (a multiple-shell model with shells of radii ranging from 1.4*r_{vdW} to 2.0*r_{vdW}, r_{vdW} is the van-der-Waals radius; U.C. Singh, P.A. Kollman, J. Comput. Chem. 5(2), 129-145 (1984)) is used if the keyword is extended:

```
$esp_fit kolman
```

12.2 Interfaces to Visualization Tools

Visualization of Molecular Geometry

The tool `t2x` can be used to convert the atomic coordinates stored in the `$grad` and `$coord` data groups into the xyz-format, which is supported by most viewers, e.g. `jmol` (`http://jmol.sourceforge.net/`). Typing

```
t2x > opt.xyz
```

in a directory containing the `control` file generates a series of frames using the information of `$grad`. Note `t2x` writes to standard output which here is redirected to a file. If you are only interested in the most recent structure, type

```
t2x -c > str.xyz
```

which only extracts the information on `$coord`.

Visualization of Densities, MOs, Electrostatic Potentials and Fields

There are several possibilities to visualize molecular orbitals or densities. `tm2molden` simply converts MO and geometry information to `molden` format. The conversion program is interactive and self-explanatory. The generated file can be visualized using either molden (`http://www.cmbi.ru.nl/molden/molden.html`) or molekel (`http://www.cscs.ch/molekel/`). For larger systems this may become very time-consuming, as plotting data (values on grids) are calculated by the respective programs (molden, molekel). It is more efficient to calculate the data for plots (MO amplitudes, densities, etc.) by `TURBOMOLE` modules and to use a visualization tool afterwards, a way, that is described in the following.

Calculation of data on grids to be used for plots with visualization tools (e.g. gOpenMol, available via `http://www.csc.fi/gopenmol/`) is driven by the keyword `$pointval`. This keyword is evaluated by all density matrix generating `TURBOMOLE`

modules, i.e. by `dscf`, `ridft`, `rimp2` `mpgrad`, `ricc2` (see Section 9.3.3) and `egrad`. Note, that all of the following quantities may be calculated simultaneusly, and that for programs `dscf`, `ridft`, `rimp2` and `mpgrad` the density matrix generating steps may be skipped by typing "`<program> -proper`".

Electron densities For the above mentioned programs setting of keyword

`$pointval dens`

or simply

`$pointval`

yields calculation of densities

$$\rho(\vec{R}_P) = \sum_{\nu\mu} D_{\nu\mu}\phi_\nu(\vec{R}_P)\phi_\mu(\vec{R}_P) \tag{12.1}$$

dens

on an orthogonal grid (R_P), the size of which is automatically adjusted to the size of the molecule and the resolution is adjusted to yield acceptable gOpenMol plots (for specification of non-default grid types (planes, lines) and non-default output formats see Section 14.2.17).

Names of output files are:

`td.plt`	total density (UHF: α density plus β density)
`sd.plt`	spin density (α density minus β density)
`mp2d.plt`	MP2 density
`mp2sd.plt`	MP2 spin density
`ed.plt`	differential density for excited state
`esd.plt`	differential spin density for excited state
`<myname>.plt`	general density passed e.g. by the `ricc2` program.

The `.plt` files may directly be visualized by gOpenMol; the file `coord.xyz`, which is also necessary for gOpenMol, is generated by the above programs, if `$pointval` is set in the `control`-file.

Two-component wave functions (only module `ridft` and only if `$soghf` is set): Total density is on file `td.plt` like for one-component wave functions; this is also true for all other quantities depending only on the density matrix (electrostatic potential etc.). `sd.plt` contains the absolute value of the spin vector density, which is the absolute value of the following vector:

$$s_i(\mathbf{r}) = \begin{pmatrix} \Psi_\alpha^* & \Psi_\beta^* \end{pmatrix} \boldsymbol{\sigma}_i \begin{pmatrix} \Psi_\alpha \\ \Psi_\beta \end{pmatrix} \qquad i = x, y, z$$

```
$pointval fmt=txt
```

leads to a file containing the spin vector density vectors, which can be used by gOpenMol. It is advisable to choose ca. one Bohr as the distance between two gridpoints.

Electrostatic potentials In an analogous way electrostatic potentials can be calculated on grids.

```
$pointval pot
```

leads to calculation of the electrostatic potential of electrons and nuclei (and external constant electric fields and point charges Q if present).

$$V(\vec{R}_P) = -\int \frac{\rho(\vec{r})}{r_{Pr}} \mathrm{d}^3\vec{r} + \sum_A \frac{Z_A}{R_{PA}} + \left(\vec{R}_P\vec{E} + \sum_Q \frac{Q}{R_{PQ}}\right) \tag{12.2}$$

In order to prevent the calculation of singularities at the positions of nuclei, for gridpoints that are closer to a nucleus than 10^{-6} a.u. the charge of the respective nucleus is omitted in the calculation of the electrostatic potential for these points. The output files are termed `tp.plt`, `sp.plt`, etc.

Electric fields (as derivatives of potentials) are calculated by

```
$pointval fld
```

The absolute values of electric fields are written to files `tf.plt`, `sf.plt`, etc. For non-default grid types and outputs that allow also for displaying of components of electric fields see Section 14.2.17.

Canonical molecular orbitals. Visualization of molecular orbitals, i.e. generation of `.plt`-files containing amplitudes of MOs i

$$A_i(\vec{R}_P) = \sum_\nu c_{i\nu}\phi_\nu(\vec{R}_P) \tag{12.3}$$

or in the two-component case

$$A_i^\Gamma(\vec{R}_P) = \sum_\nu c_{i\nu}^\Gamma\phi_\nu(\vec{R}_P) \tag{12.4}$$

with Γ as a part of the coefficient matrix (Re(α), Im(α), Re(β), Im(β)), is achieved e.g. by

```
$pointval mo 10-12,15
```

This yields amplitudes for MOs/spinors 10-12 and 15 on the default grid. The numbering of MOs refers to that you get from the first column of the output of the tool `Eiger`, the one for spinors refers to the file `EIGS`. The filenames contain the type of the irreducible representation (irrep) of the MO, the current number within this irrep and in case of UHF calculations also the spin, e.g. `2a1g_a.plt`

contains amplitudes for the second alpha-spin MO of a_{1g} type. For more-dimensional irreps columns are written to separate files, e.g. `1t2g1_a.plt`, `1t2g2_a.plt` and `1t2g3_a.plt` contain the amplitutes of the three columns of the first irrep (alpha spin) of type t_{2g}.

Two-component wavefunctions (only module `ridft` and only if `$soghf` is set): By default only the density of the chosen spinors is written in files named e.g. `10a_d.plt`. Visualization of the amplitudes of the different spinor parts is achieved e.g. by

`$pointval mo 10-12,15 minco` *real*,

where *real* is a plotting threshold that may take values between zero and one. The corresponding part Γ of the spinor (Re(α, Im(α, Re(β, Im(β)) will be written to file, if N^Γ (see below) is larger than that threshold.

$$\begin{aligned} N^\Gamma &= \mathrm{tr}(\mathbf{D}^\Gamma \mathbf{S}) \\ D^\Gamma_{\mu\nu} &= \sum_i c^{\Gamma *}_{i\nu} c^\Gamma_{i\mu} \end{aligned}$$

The filenames consist of the number of the spinor according to file `EIGS` and an additional number for the respective part Γ of the spinor (1 for Re(α), 2 for Im(α), 3 and 4 for the corresponding β-parts) e.g. `10a_4.plt` for the Im(β) of spinor 10.

Localised molecular orbitals If one has generated localized molecular orbitals (LMOs, see above) they can also be visualized.

```
$pointval lmo 3-6,8
```

as an example, leads to calculation of amplitudes for LMOs 3-6 and 8. The coefficients are read from file `lmos` (UHF: `lalp` and `lbet`), the numbering is due to the output from the localizaton section. For an UHF case this means: If you included in the localization procedure e.g. 5 α-type orbitals and 3 β-type orbitals, then, if you are interested in plotting the β-type LMOs only, you have to type

```
$pointval lmo 6-8
```

Natural molecular orbitals for two-component wavefunctions (only module `ridft` and only if `$soghf` is set): In two-component calculations it is often useful to visualize natural molecular orbitals. In contrast to one-component calculations the occupation numbers are no longer close to zero, one or two, but can take any value between zero and two. Therefor

```
$natural orbitals    file=natural
$natural orbital occupation    file=natural
```

has to be set additionally to `$soghf` (also possible via `define`).
By setting

```
$pointval nmo 9
```

in `control`-file a gOpenMol-compatible file named `nmo_9.plt` is written.

Natural atomic orbitals If one has generated natural molecular orbitals (NAOs, see above) they can be visualized with the following command in the control file:

```
$pointval nao 7-9,12
```

where the numbers of the NAOs are in the output of the population analysis.

Non-default grids are decribed in detail in Sections 14.2.17. Calculation of the above quantities at single points is needed quite often, thus an example is given here.

```
$pointval geo=point
7 5 3
0 0 7
1 2 3
```

calculates densities at points (7,5,3), (0,0,7) and (1,2,3). Output is (x,y,z, density), output file suffix is `.xyz`.

We note in passing that calculation of electrostatic potential at positions of nuclei may be used as an efficient tool to distinguish atoms of similiar atomic numbers thus providing a complement to X-Ray Structure Analysis (details see ref. [119]).

Chapter 13

Treatment of Solvation Effects with COSMO

The **Co**nductor-like **S**creening **Mo**del [120] (`cosmo`) is a continuum solvation model (CSM), where the solute molecule forms a cavity within the dielectric continuum of permittivity ε that represents the solvent. The charge distribution of the solute polarizes the dielectric medium. The response of the medium is described by the generation of screening charges on the cavity surface.

CSMs usually require the solution of the rather complicated boundary conditions for a dielectric in order to obtain the screening charges. `cosmo` instead uses the much simpler boundary condition of vanishing electrostatic potential for a conductor,

$$\mathbf{\Phi}^{tot} = 0.$$

This represents an electrostatically ideal solvent with $\varepsilon = \infty$. The vector of the total electrostatic potential on the cavity surface segments is determined by the solute potential $\mathbf{\Phi}^{sol}$, which consist of the electronic and the nuclear part, and the vector of the screening charges $\mathbf{q}$,

$$\mathbf{\Phi}^{tot} = \mathbf{\Phi}^{sol} + \mathbf{A}\mathbf{q} = 0.$$

$\mathbf{A}$ is the Coulomb matrix of the screening charge interactions. For a conductor, the boundary condition $\mathbf{\Phi}^{tot} = 0$ defines the screening charges as

$$\mathbf{q} = -\mathbf{A}^{-1}\mathbf{\Phi}^{sol}.$$

To take into account the finite permittivity of real solvents, the screening charges are scaled by a factor.

$$\begin{aligned} f(\varepsilon) &= \frac{\varepsilon - 1}{\varepsilon + \frac{1}{2}} \\ \mathbf{q}^{\star} &= f(\varepsilon)\mathbf{q} \end{aligned}$$

The deviation between the `cosmo` approximation and the exact solution is rather small. For strong dielectrics like water it is less than 1%, while for non-polar solvents with $\varepsilon \approx 2$ it may reach 10% of the total screening effects. However, for weak dielectrics, screening effects are small and the absolute error therefore typically amounts to less than one kcal/mol. The dielectric energy, i.e. the free electrostatic energy gained by the solvation process, is half of the solute-solvent interaction energy.

$$E_{diel} = \frac{1}{2} f(\varepsilon) \mathbf{q}^{\dagger} \mathbf{\Phi}^{sol}$$

The total free energy of the solvated molecule is the sum of the energy of the isolated system calculated with the solvated wave function and the dielectric energy

$$E = E(\Psi^{solv}) + E_{diel}.$$

A `cosmo` energy calculation starts with the construction of the cavity surface grid. Within the SCF procedure, the screening charges are calculated in every cycle and the potential generated by these charges is included into the Hamiltonian. This ensures a variational optimization of both the molecular orbitals and the screening charges and allows the evaluation of analytic gradients.

Radii based Cavity Construction: In order to ensure a sufficiently accurate and efficient segmentation of the molecular shaped cavity the COSMO implementation uses a double grid approach and segments of hexagonal, pentagonal, and triangular shape. The cavity construction starts with a union of spheres of radii $R_i + RSOLV$ for all atoms i. In order to avoid problems with symmetric species, the cavity construction uses de-symmetrized coordinates. The coordinates are slightly distorted with a co-sinus function of amplitude AMPRAN and a phase shift PHSRAN. Initially a basis grid with NPPA segments per atom is projected onto atomic spheres of radii $R_i + RSOLV$. In order to avoid the generation of points in the problematic intersections, all remaining points, which are not in the interior of another sphere, are projected downwards onto the radius R_i. In the next step a segment grid of NSPH segments per H atom and NSPA segments for the other atoms is projected onto the surface defined by R_i. The basis grid points are associated to the nearest segment grid centers and the segment coordinates are re-defined as the center of area of their associated basis grid points, while the segment area is the sum of the basis grid areas. Segments without basis grid points are discarded. In order to ensure nearest neighbor association for the new centers, this procedure is repeated once. At the end of the cavity construction the intersection seams of the spheres are paved with individual segments, which do not hold associated basis grid points.

Density based Cavity Construction: Instead of using atom specific radii the cavity can be defined by the electron density. In such an isodensity cavity construction one can use the same density value for all atoms types or the so-called scaled isodensity values. In the later approach different densities are used for the different atom types. The algorithm implemented in TURBOMOLE uses a marching

tetrahedron algorithm for the density based cavity construction. In order to assure a smooth density change in the intersection seams of atoms with different isodensity specification, this areas are smoothend by a radii based procedure.

A-Matrix Setup: The **A** matrix elements are calculated as the sum of the contributions of the associated basis grid points of the segments k and l if their distance is below a certain threshold, the centers of the segments are used otherwise. For all segments that do not have associated basis grid points, i.e. intersection seam segments, the segment centers are used. The diagonal elements A_{kk} that represent the self-energy of the segment are calculated via the basis grid points contributions, or by using the segment area $A_{kk} \approx 3.8\sqrt{a_k}$, if no associated basis grid points exist.

Outlying charge correction: The part of the electron density reaching outside the cavity causes an inconsistency that can be compensated by the "outlying charge correction". This correction will be performed at the end of a converged SCF or an iterative MP2 calculation and uses an outer surface for the estimation of the energy and charge correction [121]. The outer surface is constructed by an outward projection of the spherical part of the surface onto the radius $R_i + ROUTF * RSOLV$. It is recommended to use the corrected values.

Numerical Frequency Calculation: The calculation of harmonic frequencies raises the problem of non-equilibrium solvation in the `cosmo` framework, because the molecular vibrations are on a time scale that do not allow a re-orientation of the solvent molecules. Therefore, the total response of the continuum is split into a fast contribution, described by the electronic polarization, and a slow term related to the orientational relaxation. As can be shown [122] the dielectric energy for the disturbed state can be written as

$$E^d_{diel} = \frac{1}{2} f(\varepsilon)\mathbf{q}(\mathbf{P}^0)\mathbf{\Phi}(\mathbf{P}^0) + \frac{1}{2} f(n^2)\mathbf{q}(\mathbf{P}^\Delta)\mathbf{\Phi}(\mathbf{P}^\Delta) + f(\varepsilon)\mathbf{q}(\mathbf{P}^0)\mathbf{\Phi}(\mathbf{P}^\Delta),$$

where $\mathbf{P}^\Delta$ denotes the density difference between the distorted state and the initial state with density $\mathbf{P}^0$. The interaction is composed of three contributions: the initial state dielectric energy, the interaction of the potential difference with the initial state charges, and the electronic screening energy that results from the density difference. The energy expression can be used to derive the correspondent gradients, which can be applied in a numerical frequency calculation. Because the `cosmo` cavity changes for every distorted geometry the initial state potential has to be mapped onto the new cavity in every step. The mapped potential of a segment of the new cavity is calculated from the distance-weighted potentials of all segments of the old cavity that fulfill a certain distance criterion. The mapped initial state screening charges are re-calculated from the new potential.

Iterative MP2 COSMO: For ab initio MP2 calculations within the CSM framework three alternatives can be found in the literature [123]. The first approach, often

referred to as PTE, performs a normal MP2 energy calculation on the solvated HF wave function. The response of the solvent, also called reaction field, is still on the HF level. It is the only of the three approaches that is formally consistent in the sense of second-order perturbation theory [124, 125]. In the so-called PTD approach the vacuum MP2 density is used to calculate the reaction field. The third approach, often called PTED, is iterative so that the reaction field reflects the density of the first-order wave function. In contrast to the PTE approach the reaction field, i.e. the screening charges, change during the iterations until self consistency is reached. Gradients are available on the formally consistent PTE level only [126].

Chapter 14

Keywords in the control file

14.1 Introduction

The file control is the input file for TURBOMOLE which directly or by cross references provides the information necessary for all kinds of runs and tasks. control is usually generated by define, the input generator. This chapter provides a short-hand documentation: a list of the most important key words, the possible parameters for each keyword, default values, and a brief explanation.

14.2 Format of Keywords and Comments

TURBOMOLE input is keyword-directed. Keywords start with a '$', e.g. $title. Comments may be given after $dummy, or by a line starting with #; these lines are ignored by TURBOMOLE. Blank lines are also ignored. Keywords may be in any order unless stated otherwise below.

The sample inputs given below should help to give an idea how the keywords are to be used. They are sorted according to program. Complete control files are provided in Chapter 15. An alphabetical list of all keywords is given in the index.

14.2.1 General Keywords

```
$operating system unix
$path
$lock off
$suspend off
```

The four keywords above are set by define, but are not necessary.

```
$statistics dscf
```
or
```
$statistics mpgrad
```

Only a statistics run will be performed to determine file space requirements as specified for `dscf` or `mpgrad`. On return the statistics option will be changed to `$statistics off`.

```
$actual step dscf
```

means *current step*. Keyword and data group (as e.g. `dscf`) is set by every program and removed on successful completion.

```
$last step relax
```

Keyword and data group (as e.g. `relax`) set by every program on successful completion.

General file cross-references:

```
$coord              file=coord
$intdef             file=coord
$user-defined bonds file=coord
$basis         file=basis
$ecp           file=basis
$jbas          file=auxbasis
$scfmo           file=mos
$uhfmo_alpha     file=alpha
$uhfmo_beta      file=beta
$natural orbitals           file=natural
$natural orbital occupation file=natural
$energy        file=energy
$grad          file=gradient
$forceapprox   file=forceapprox
```

It is convenient not to include all input in the `control` file directly and to refer instead to other files providing the corresponding information. The above cross references are default settings from `define`; you may use other file names. `define` will create most of these files. Examples of these files are given below in the samples.

`$coord` (and `$intdef` and `$userdefined bonds`)
: contains atom specification—type and location—and the bonds and internal coordinates convenient for geometry optimizations.

`$basis`
: specification of basis sets.

`$ecp` specification of effective core potentials.

`$jbas`
auxiliary (fitting) basis for the Coulomb terms in `ridft`.

`$scfmo`, `$uhfmo_alpha`, `$uhfmo_beta`
MO vectors of SCF or DFT calculations for RHF or UHF runs.

`$natural orbitals`, `$natural orbital occupation`
keywords and data groups set by unrestricted `dscf` or `ridft` runs. Contain natural MO vector and orbital occupation.

`$energy`, `$grad`
energies and gradients of all runs, e.g. for documentation in a geometry optimizations.

`$forceapprox`
approximate force constant for geometry optimizations.

The `control` file must end with this keyword:

`$end`

14.2.2 Keywords for System Specification

General information defining the molecular system: nuclear coordinates, symmetry, basis functions, number of occupied MOs, etc. which are required by every module.

`$title`
give title of run or project here.

`$symmetry d4h`
Schönflies symbol of the point group. All point groups are supported with the exception of NMR shielding and force constant calculations etc. which do not work for groups with complex irreps (C_3, C_3h, T, etc). Use a lower symmetry group in this case.

`$atoms`
Example:

```
$atoms
cu 1-4                                                    \
   basis =cu ecp-18 arep                                  \
   jbas  =cu ecp-18                                       \
   ecp   =cu ecp-18 arep
se 5-6                                                    \
   basis =se ecp-28 arep dzp                              \
   jbas  =se ecp-28                                       \
   ecp   =se arep
```

note the backslash \ : this is necessary. For each type of atom, one has to specify

- the basis set
- and the auxiliary (fitting) basis for RIDFT calculations
- the ECP if this is used.

The files `basis`, `ecp` and `jbas` must provide the necessary information under the labels specified in `$atoms`.

`$pople` *char*
This data group specifies the number of cartesian components of basis functions (i.e. 5d and 7f in AO-Basis, 6d and 10f in CAO-Basis) for which the SCF calculation should be performed. Possible values for *char* are *AO* (default) or *CAO*. If CAO is used—which is not recommended—a core guess must be used instead of a Hückel guess (see `$scfmo`).

RHF

`$closed shells`
Specification of MO occupation for RHF, e.g.

```
 a1g     1-4                                    ( 2 )
 a2g     1                                      ( 2 )
```

`$open shells type=1`
MO occupation of open shells and number of open shells. `type=1` here means that there is only a single open shell consisting e.g. of two MOs:

```
b2g     1                                      ( 1 )
b3g     1                                      ( 1 )

$roothaan          1
a = 1      b = 2
```

`$roothaan`
Roothaan parameters for the open shell, here a triplet case. `define` recognizes most cases and suggests good Roothaan parameters.

For further information on ROHF calculations, see the sample input in Section 15.6 and the tables of Roothaan parameters in Section 6.3.

UHF

`$uhf` directs the program to carry out a UHF run,e.g.

```
$alpha shells
 a1g     1-4                                    ( 1 )
 a2g     1                                      ( 1 )
$beta shells
 a1g     1-4                                    ( 1 )
 a2g     1                                      ( 1 )
```

The specification of MO occupation for UHF, `$uhf` overwrites closed-shell occupation specification.

14.2.3 Keywords for redundant internal coordinates in `$redund_inp`

With the parameters in `$redund_inp` the generation of redundant internal coordinates can be modified. All entries have to be made in the `control` file before invoking the `ired` option. Important options are:

`iprint` n
: print parameter for debug output: The larger n is, the more output is printed $n \geq 0, n \leq 5$ (default: 0)

`metric` n
: method for generating and processing of redundant internal coordinates $n \geq -3, n \leq 3, n \neq 0$ (default: 3)

Values for the `metric` option:

`n = 1` "Delocalized Coordinates"
: The $\mathbf{BmB^t}$ matrix is diagonalized for the complete set of redundant internal coordinates, matrix $\mathbf{m}$ is a unit matrix.

`n = -3` Delocalized Coordinates obtained with a modified matrix $\mathbf{m}$, the values of $\mathbf{m}$ can be defined by user input (see below).

`n = -1` "Hybrid Coordinates"
: Natural internal coordinates are defined as in the old `iaut` option. If a cage remains, delocalized coordinates (as for n=1) are defined for the cage.

`n = -2` Very simular to the $n = 1$ option, but for the remaining cage delocalized coordinates with modified matrix $\mathbf{m}$ are defined as for $n = -3$.

`n = 2` "Decoupled coordinates"
The redundant coordinates are divided into a sequence of blocks. These are expected to have decreasing average force constants, i.e. **stretches, angle coordinates, torsions and "weak" coordinates**. The $\mathbf{BB^t}$ matrix is diagonalized for each block separately after the columns of **B** were orthogonalized against the columns of **B** of the the preceding blocks.

`n = 3` "Generalized natural coordinates"
Natural internal coordinates are defined first, for the remaining cage decoupled coordinates are defined.

`type` r
a positive real number, which is an approximate "force constant", can be read in for each type of coordinate (see below). The force constants are used for the definition of the matrix **m** in $\mathbf{BmB^t}$.

Types of internal coordinates for the definition of m

The matrix **m** is assumed to be a diagonal matrix. For each type of coordinate a different value for the force constants m_{ii} can be read in. Types of coordinates are:

`stre` bond stretch (default: 0.5)

`invr` inverse bond stretch (default: 0.5)

`bend` bond angle (default: 0.2)

`outp` Out of plane angle (default: 0.2)

`tors` dihedral or "torsional" angle (default: 0.2)

`linc` Special angle coordinate for collinear chains, bending of the chain a–b–c in the plane of b–c–d (default: 0.2)

`linp` bending of the chain a–b–c perpendicular to the plane of b–c–d (default: 0.2)

`wstr` stretch of a "weak" bond, i.e. the bond is assumed to have a very low force constant, e.g. a "hydrogen bond" or a "van der Waals bond" (default: 0.05)

`winv` inverse stretch of a weak bond (default: 0.05)

`wbnd` bond angle involving at least one weak bond (default: 0.02)

`wout` Out of plane angle for weak bonds (default: 0.02)

`wtor` dihedral angle for weak bonds (default: 0.02)

`wlnc` linc coordinate for weak bonds (default: 0.02)

`wlnp` linp coordinate for weak bonds (default: 0.02)

14.2.4 Keywords for Module UFF

One has to specify only the cartesian coordinates (data group `$coord`) to start a `uff` run. The program `uff` requires the data groups `$uff`, `$ufftopology`, `$uffgradient` and `$uffhessian`. If these keywords do not exist in the `control` file the program will generate these data groups.

The data group `$uff` contains the parameters described below. The default values—in the `control` file—are:

```
      1         1          0 ! maxcycle,modus,nqeq
 111111                     ! iterm
0.10D-07  0.10D-04          ! econv,gconv
    0.00  1.10              ! qtot,dfac
0.10D+03  0.10D-04    0.30 ! epssteep,epssearch,dqmax
      25      0.10    0.00 ! mxls,dhls,ahls
    1.00      0.00    0.00 ! alpha,beta,gamma
       F         F       F ! transform,lnumhess,lmd
```

The explanation of the variables are as follows:

`maxcycle`
: number of max. optimization cycles (`maxcycle=1`: single-point calculation).

`modus`
: can have the values `+1` or `-1`. If `modus = -1` only the topology will be calculated.

`nqeq` each `nqeq` cycle the partial charges will be calculated. If `nqeq = 0`, then the partial charges are calculated only in the first cycle, if the file `ufftopology` does not exist.

`iterm`
: switch for the different types of force field terms:

`100000`	bond terms will be calculated.
`010000`	angle terms will be calculated.
`001000`	torsion terms will be calculated.
`000100`	inversion terms will be calculated.
`000010`	non bonded van der Waals terms will be calculated.
`000001`	non bonded electrostatic terms will be calculated.

`econv`, `gconv`
: convergence criteria for energy and gradient.

`qtot` total charge of the molecule.

`dfac` distance parameter to calculate the topology. If the distance between the atoms I and J is less than the sum of the covalent radii of the the atoms multiplied with `dfac`, then there is a bond between I and J.

`epssteep`
if the norm of the gradient is greater than `epssteep`, a deepest-descent-step will be done.

`epssearch`
if the norm of the gradient is smaller than `epssearch`, no linesearch step will be done after the Newton atep.

`dqmax`
max. displacement in a.u. for a coordinate in a relax step.

`mxls`, `dhls`, `ahls`
parameters of linesearch:

`ahls`	start value
`dhls`	increment
`mxls`	number of energy calculations

`alpha`, `beta`, `gamma`
modification parameter for the eigenvalues of the Hessian (see below): $f(x) = x * (\texttt{alpha} + \texttt{beta} * \exp(-\texttt{gamma} * x))$.

`transform`
a switch for the transformation in the principal axis system.

`lnumhess`
a switch for the numerical Hessian.

`lmd` a switch for a MD calculation.

Input Data Blocks Needed by UFF

`$coord`
cartesian coordinates of the atoms (default: `$coord file=coord`)

`$ufftopology`
contains a list of the next neigbours of each atom (see Section 14.2.4). Sometimes it is useful to enter the connectivity (in the input block `nxtnei12` in the file `ufftopology`) by hand (not always necessary; default: `$ufftopology file=ufftopology`).

Beyond this `uff` reads the force field parameters for the atoms from the file `parms.in`. If this file exists in the directory from which one starts an `uff` calculation the program will use this file, if not the program reads the data from the file `$TURBODIR/uff/parms.in`.

If one wants own atom types, one has to add these atoms types in the file `parms.in`. For each new atom type one has to specify the *natural* bond distance, the *natural* bond angle, the *natural* non-bond distance, the well depth of the Lennard-Jones potential, the scaling factor ζ, the effective charge, torsional barriers invoking a pair of sp^3 atoms, torsional barriers involving a pair of sp^2 atoms, generalized Mulliken–Pauling electronegativities, the idem potentials, characteristic atomic size, lower bound of the partial charge, upper bound of the partial charge. Distances, energies and charges are in atomic units and angles are in rad.

UFF Output Data Blocks

`$coord`
: contains the (updated) cartesian coordinates of the atoms (default: `$coord file=coord`).

`$ufftopology`
: contains the full information of the topology of the molecule and the whole force field terms (see below; default: `$ufftopology file=ufftopology`).

`$uffgradient`
: contains the accumulated cartesian analytical gradients (default: `$uffgradient file=uffgradient`).

`$uffhessian`
: contains the cartesian analytical Hessian;
(default: `$uffhessian file=uffhessian0-0`).

The file `ufftopology`

The topology file `ufftopology` contains the blocks nxtnei12, nxtenei13, nxtnei14, connectivity, angle, torsion, inversion, nonbond and qpartial. It starts with `$ufftopology` and ends with `$end`. The first three blocks (nxtnei12, nxtnei13, nxtnei14) have the same form: they start with the atom number and the number of its neighbours, in the next line are the numbers of the neighbour atoms. Then the *connectivity*-block follows starting with the number of bond terms. Each line contains one bond term:

$$I \quad J \quad \mathrm{d} \quad \mathrm{BO}.$$

Here are I and J the number of the atoms, d the distance in a.u. and BO is the bond order.

The angle terms follow, starting with the number of the angle terms. In each line is one angle term:

$$J \quad I \quad K \quad \mathrm{wtyp} \quad \theta \quad nr_{JI} \quad nr_{IK}.$$

Here are J, I and K the atoms number, where atom I is in the apex. "wtyp" is the angle type and has the following values:

wtyp = 1 linear case

wtyp = 2 trigonal planar case

wtyp = 3 quadratic planar case

wtyp = 6 octahedral case

wtyp = 9 all other cases.

θ is the angle value in degree. nr_{JI} and nr_{IK} are the number of the bonds between J and I and the bond between I and K. The hybridization of atom I determines "wtyp".

Then the torsion terms follow, starting with the number of the torsion terms. Each line contains one torsion term:

$$I \quad J \quad K \quad L \quad nr_{JK} \quad \text{ttyp} \quad \phi \quad \theta_{IJK} \quad \theta_{JKL}.$$

Here are I, J, K and L the atom numbers. nr_{JK} is the number of the bond between J and K. "ttyp" is the torsion type:

ttyp = 1 J (sp^3)–K (sp^3)

ttyp = 11 like ttyp=1, but one or both atoms are in Group 16

ttyp = 2 J (sp^2)–K (sp^3) or vice versa

ttyp = 21 like ttyp=2, but one or both atoms are in Group 16

ttyp = 22 like ttyp=2, but J or K is next a sp^2 atom

ttyp = 3 J (sp^2)–K (sp^2)

ttyp = 9 all other cases.

ϕ is the value of the torsion angle in degree. θ_{IJK} is the angle value of $(I - J - K)$ and θ_{JKL} is the cwone for $J - K - L$. The hybridizations of J and K determine "ttyp".

The inversion terms follow starting with the number of inversion terms (e.g. the pyramidalisation of NH_3). In each line is one inversion term:

$$I \quad J \quad K \quad L \quad \text{ityp1} \quad \text{ityp2} \quad \text{ityp3} \quad \omega_1 \quad \omega_2 \quad \omega_3.$$

I, J, K and L are the atom numbers. Atom I is the central one. ityp1, ityp2, ityp3 are the types of the inversions:

ityp = 10 atom I is C and atom L is O

ityp = 11 like ityp=10, but L is any atom

ityp = 2 I is P

ityp = 3 I is As

ityp = 4 I is Sb

ityp = 5 I is Bi

ityp = 9 all other cases.

ω_1, ω_2 and ω_3 are the values of the inversion angles in degree.

The nonbond terms follow starting with the number of the non-bonded terms. In each line is one nonbond term:

$$I \qquad J \qquad \mathrm{d}\ .$$

Here I and J are the atom numbers, d the distance in a.u. Then the partial charges follow.

If the determination of the molecule connectivity failed, you can specify the block nxtnei12 in the file `ufftopology`. Then the program calculates the other blocks.

Based on the numbers of the next neighbours (block nxtnei12 in the file `ufftopology`) the program tries to determine the UFF type of an atom. The following rules are implemented: If the atom has three next neighbours and it is in the nitrogen group, then it has a hybridization three. If it is not in the nitrogen group, it has hybridization two. If the atom has four next neighbours and it is in the carbon group, it has hybridization three. If it is not in the carbon group, it becomes hybridization four. If the number of next neighbours is six, then it gets the hybridization six.

Since the smallest eigenvalues λ_i of the Hessian has the greatest influence on the convergence of the geometry optimization, one can shift these values with

$$\tilde{\lambda}_i = \lambda_i \cdot \left(\alpha + \beta \cdot e^{-\gamma x}\right)$$

and calculates a new Hessian with these modified eigenvalues.

14.2.5 Keywords for Modules DSCF and RIDFT

`$denconv` *real*

Convergency criterion for the root mean square of the density matrix. If you want to calculate an analytical MP2 gradient (program `mpgrad`) *real* should be 1.d-7 or less.

`$dft` *options*

Listing of all possible sub-keywords for `$dft` (cross-references are given).

The user normally has to choose only the functional and the grid size, see below. All other parameters have proven defaults.

`functional` *name*

Specification of the functional, default is BP86, printed as `functional`

`b-p`. For all possible—and useful—functionals, please refer to page 228 and for definition of the functionals the section 6.2 on page 114.

Example (default input):

```
$dft
   functional b-p
```

`gridsize` *integer* or `m`*integer*

Specification of the spherical grid (see section 14.2.5 on page 228). Default is `gridsize m3`.

Example:

```
$dft
   gridsize m3
```

`gridtype` *integer* —not recommended for use—

Specification of the mapping of the radial grid.

Possible values for `gridtype` are $1, \ldots, 6$, but `gridtype 4` to `6` is only for the use with `functional lhf` (see page 231). For the definition of `gridtype 1` – `3`, please refer to Eq. (16), (17) and, (19) in Ref. [127].

Example (default value):

```
$dft
   gridtype 3
```

`debug` *integer*

Flag for debugging. `debug 0` means no debug output (default), `debug 1` means some output, `debug 2` means a lot more output. Be careful!

`nkk` *integer*

Specification of the sharpness of the partition function as proposed by Becke [128], default is `nkk 3`. The usage of `nkk` makes sense only in the range $1 \leq$ `nkk` ≤ 6.

Example:

```
$dft
   nkk 3
```

`ntheta` *integer*
`nphi` *integer* —not recommended for use—

Only for user-specified Lobatto grids, i.e. `gridsize 9`, `ntheta` specifies the number of θ points and `nphi` specifies the number of ϕ points. For the fixed Lobatto grid, i.e. `gridtype 8`, the default value is `ntheta 25` and `nphi 48`.

When `gridsize 9` is given, you have to specify both, `ntheta` and `nphi` (see below), otherwise the program will crash. The restriction for user-defined Lobatto grids is the number of grid points, which must not exceed 2000 grid points.

Example:

```
$dft
  gridsize 9
  ntheta 30
  nphi 60
```

`old_RbCs_xi`

Original grids had not been carefully optimized for all atoms individually. This has now been done, which let to changes of ξ for Rb and Cs only resulting in minor improvements. If you have ongoing projects, which have been started with the old grids, you should continue using them with the keyword `old_RbCs_xi`.

Example:

```
$dft
   old_RbCs_xi
```

`radsize` *integer*

Specifies the number of radial grid points. Default values depend on type of atom and grid (see keyword `gridsize`). The formula for the radial gridsize is given as,

$$\text{number of radial grid points} = \text{ioffrad} + (\texttt{radsize} - 1) * 5\,.$$

ioffrad is atom-dependent, the more shells of electrons, the larger ioffrad:

elements	ioffrad	elements	ioffrad
for H–He	20	for K–Kr	40
for Li–Ne	25	for Rb–Xe	45
for Na–Ar	30	for Cs–Lw	50

The radial grid size increases further for finer grids:

`gridsize`	1	2	3	4	5	6	7	8	9
`radsize`	1	2	3	6	8	10	14	9	3

If you want to converge results with respect to radial grid size, increase `radsize` in steps of 5, which is convenient (see equation above).

`diffuse` *integer*

Serves to increase quadrature grids; this is recommended in case of very diffuse wavefunctions. With the keyword `diffuse` grids are modified by changing the linear scaling factor ξ of radial grid points and the radial gridsize:

`radsize` $\Longrightarrow$ `radsize` + *incr*
$\xi \Longrightarrow \xi$ * *scal*

`diffuse` *integer*	1	2	3	4	5	6
incr	1	2	3	4	5	6
scal	1.5	2.0	2.8	4.0	5.0	6.0

For information about the linear scaling parameter ξ, see Eq. (16)–(19) and Table 1 in Ref. [127].

In addition, the reduction of spherical grid points near nuclei is supressed, i.e. `fullshell on` is set (see page 217).

Note: the keyword `radsize` *integer* overrules the setting of *incr*; for more information, see p. 215.

Recommendation: For diffuse cases use `gridsize m4` (or larger) in combination with `diffuse 2` and check the number of electrons; for more difficult cases use `diffuse 4`. In case of doubt, verify the calculation with a larger grid, i.e. `gridsize 7`.

The test suite example `$TURBODIR/TURBOTEST/dscf/short/H3PO4.DSCF.DIFFUSE` provides an example of usage; this also gives reasonable values for damping and orbitalshift to reach convergence in this and similar cases, see `$scfdamp` and `$scforbitalshift` (p. 222 and p. 225).

Example (Recommendation):

```
$dft
  gridsize m4
  diffuse 2
```

`rhostart` *integer*
`rhostop` *integer* —for developers only—

Radial grid points have a linear scaling parameter ξ, see Eq.(16)–(19) and Table 1 in Ref. [127]. With the following input,

```
$dft
   rhostart 50
   rhostop 200
```

one performs a numerical integration for the density and the exchange correlation term for $\xi = 0.5, (0.01), 2.0$ for given MOs and functional. NOTE: only molecules with a single atom type can be used. The results serve to establish stable, optimal ξ values, see Figure 1 in Ref. [127]. Program stops after this testing.

`reference`

Usage of the reference grid, which is a very fine grid with very tight thresholds. The default values for the different variables are:

```
gridsize 7
radsize 14
fullshell 1
dgrenze 16
```

```
fgrenze 16
qgrenze 16
fcut 16
```

Please refer to the corresponding sub-keywords for explanation.

If you want to use the reference grid, you have to skip the keyword `gridsize`, and type instead `reference`. Example:

```
$dft
   functional b-p
   reference
```

`test-integ` —only with `functional lhf`—
Checking if the selected grid is accurate enough for the employed basis-set by performing a numerical integration of the norm of all orbitals. For more detailed information about LHF, please refer to page 231.

`batchsize` *integer*
Grid points are sorted into batches, which are then processed. This increases efficency. This should be changed only by developers. Default is `batchsize 100`.

`fullshell`
Standard grids have reduced number of spherical grid points near nuclei. With the keyword `fullshell` this reduction is suppressed. Reference grid (see keyword `reference`) always has full spherical grids with 1202 points. Should be used to checked the influence of spherical grid reduction.

Example for the usage of `fullshell`:

```
$dft
   functional b-p
   gridsize m4
   fullshell
```

`symblock1` *real*
`symblock2` *real* —for developers only—
Values of real effects efficiency of the quadrature, default is `symblock1 0.001` and `symblock2 0.001`, one can try higher or smaller values.

`xparameter` *integer* —not recommended for use—
Where `xparameter` (default) can be: `sgrenze` (8), `fgrenze` (10), `qgrenze` (12), `dgrenze` (12) and, `fcut` (14). These parameters control neglect of near zeros of various quantities. With `xparameter` *integer* one changes the default. *integer* larger than defaults will increase the numerical accuracy. Tighter threshold are set automatically with keyword `$scfconv` (see section 14.2.5 on page 222).

`weight derivatives`
: Includes the derivatives of quadrature weights to get more accurate results. Default is that the derivatives of quadrature weights will be not considered, see section 14.2.8 on page 240.

`gridordering`
: Grid points are ordered into batches of neighbouring points. This increases efficiency, since now zeros can be skipped for entire batches. `gridordering` is default for serial version, not for the parallel one. You cannot use `weight derivatives` and `gridordering` together.

  Example for switching off `gridordering`:

```
$dft
   gridordering 0
```

`$electrostatic field`
: Specification of external electrostatic field(s). The specification may take place either by `Ex, Ey, Ez` or by `x, y, z, |E|`. See also `$fldopt`.

  Example:

```
$electrostatic field
       0.1000E-03   0.000       0.000
```

`$fermi tmstrt=<300.0> tmend=<100.0> tmfac=<0.9> hlcrt=<1.0E-01> stop=<1.0E-03> nue=<N>`

Requests calculation of occupation numbers at a finite temperature T. For an orbital with the energy ε_i the occupation number $n_i \in [0, 1]$ is calculated as

$$n_i = \frac{1}{2}\mathrm{erfc}\left(\frac{\varepsilon_i - \mu}{fT}\right),$$

where μ is the Fermi level. The factor $f = 4k/\sqrt{\pi}$ is chosen to yield the same slope at the Fermi level as the Fermi distribution.

Calculation of the fractional occupation numbers starts when the current HOMO-LUMO gap drops below the value given by `hlcrit` (default: 0.1). The initial temperature given by `tmstrt` (default: 300 K) is reduced at each SCF cycle by the factor `tmfac` (default: 1.0) until it reaches the value `tmend` (default: 300 K). Note that the default values lead to occupation numbers calculated at a constant $T = 300$ K. Current occupation numbers are frozen if the energy change drops below the value given by `stop` (default: $1{\cdot}10^{-3}$). This prevents oscillations at the end of the SCF procedure.

Calculation of fractional occupation numbers often requires much higher damping and orbital shifting. Please adjust the values for `$scfdamp` and `$scforbitalshift` if you encounter convergence problems.

In UHF runs this option can be used to automatically locate the lowest spin state. In order to obtain integer occupation numbers `tmend` should be set to relatively low value, e.g. 50 K.

Calculation of fractional occupation numbers should be used only for single point calculations. When used during structure optimizations it may lead to energy oscillations.

The optional `nue` value (number of unpaired electrons) can be used to force a certain multiplicity in case of an unrestricted calculation. `nue=0` is for singlet, `nue=1` for dublet, etc.

`$firstorder`

Perform first-order SCF-calculation, i.e. perform only one SCF-iteration with the start MOs (which should be the orthogonalized MOs of two independent subsystems as is explained in detail in Chapter 12).

`$fldopt` *options*

Specification of options related with external electrostatic fields. The following options are available:

`1st derivative on/off`

Calculate numerical 1st derivative of SCF energy with respect to electrostatic field (default: off), increment for numerical differentiation is `edelt` (see below).

`2nd derivative on/off`

Calculate numerical 2nd derivative of SCF energy with respect to electrostatic field (default: off), increment for numerical differentiation is edelt.

`edelt=` *real*

Increment for numerical differentiation (default: 0.005).

`fields on/off`

Calculate SCF energy for non-zero external electrostatic fields defined in `$electrostatic field`.

`geofield on/off`

Calculate SCF energy for one external field definition and dump disturbed MOs onto `$scfmo`. This enables to evaluate properties or perform geometry optimizations in the presence of an external field.

Caution: don't use the RI approximation for all these calculations since this will lead to non-negligible errors!!

`$incore` *integer*

By using this option the two-electron integrals are kept in RAM; *integer* specifies how many megabytes should be allocated. If the integrals exceed the RAM allocated the program reverts to the standard mode. Supports all methods which process two-electron integrals, i.e. SCF and DFT (including hybrid functionals); RHF and UHF.

The following condition must be met:

```
$scfdenapproxl 1
```

and rhfshells 1 or 2. It is advisable to set `$thize` as small as possible (e.g. `$thize 0.1d-08`) and to remove the keyword `$scfdump`.

Note: this keyword does not work for parallel runs.

`$mo-diagram` *only* `nirreps=`*integer*

If this keyword is set the energies and symmetry labels of all occupied MOs will be dumped to this data group. This may be helpful to draw mo-diagrams. If *only* has been set only the start MOs are dumped and the program quits.

`nirreps` will hold the total number of displayed orbitals after the successful run.

`$moprint`

If this keyword is present all occupied orbitals are dumped to standard output. Be careful about this option as it can create huge output files in case of many basis functions.

`$mo output format` *format*

If this line is present, the dscf program is forced to output the MOs using the new FORTRAN format *format* regardless of the *format*-option in data group `$scfmo`. Otherwise the input format will be used.

Example: `$mo output format(3(2x,d15.8))`

`$natural orbitals`

This data group will be written after an UHF calculation (together with $natural orbital occupation) and contains the natural space orbitals (same syntax as `$scfmo`).

`$natural orbital occupation`

This data group will be written after an UHF calculation (together with `$natural orbitals`) and contains the occupation of natural orbitals (syntax as any data group related with orbital occupation information, e.g. `$closed shells`), e.g.

```
 a       1-5                    (     2.00000000000000 )
 a       6                      (     1.99949836999366 )
 a       7                      (     1.99687490286069 )
 a       8                      (     1.00000000000000 )
 a       9                      (      .00312509713931 )
 a       10                     (      .00050163000634 )
```

`$point_charges`

Specification of position and magnitude of point charges to be included in the Hamiltonian. Each point charge is defined in the format

```
<x> <y> <z> <q>
```

with <x>, <y>, <z> being the coordinates and <q> its charge,e.g.

```
$point_charges thr=<real> self-energy list
   2. 2. 2. 5.
   5. 0. 0. 2.5
```

In addition the following optional arguments may be given:

thrdistance threshold for discarding redundant point charges, default value 10^{-6}.

selfenergyif given, the selfenergy of the point charge array will will be included in the energy and the gradient

listprint all point charges in the output (default is to print the point charges only if less than 100 charges given)

$prediag

concerns the first SCF iteration cycle if start MOs from an EHT guess are used.

The SCF iteration procedure requires control mechanisms to ensure (fast) convergence, in TURBOMOLE these are based on orbital energies ϵ_i of the preceeding iteration used for level shifting and damping (besides DIIS, see below). This feature cannot be used in the first iteration if EHT MOs are employed as start, since ϵ_i are not available. The keyword $prediag provides '$\epsilon_i$ of the zeroth iteration' by diagonalization of occ–occ and virt–virt part of the first Fock matrix, to allow for level shifting etc.. See $scfdiis below.

$restart dscf *twoint*

Try a dscf restart. The program will read the data group $restartd (which must exist, also $scfmo has to exist!) and continue the calculation at the point where it ended before. If the additional option twoint is appended, the program will read the two-electron integrals from the files specified in $scfintunit, so there will be almost no loss of cpu-time.

All this information is normally provided by the previous dscf run if the keyword $scfdump (see there) was given.

$restartd *data*

Data provided by a previous dscf run that has been interrupted. This keyword is created when $scfdump was given.

$rundimensions *data*

is set by define so usually no changes are necessary. The dimensions must be greater or equal to those actually required, i.e. you can delete basis functions and keep rundimensions. This keyword is not necessary for small cases.
Example:

```
dim(fock,dens)=6072
natoms=6
nshell=34
nbf(CAO)=108
nbf(AO)=98
dim(trafo[SAO<-->AO/CAO])=256
rhfshells=1
```

`$scfconv` *integer*
SCF convergency criterion will be $10^{-integer}$ for the energy. Gradients will only be evaluated if *integer* > 6.

`$scfdamp start=<.500> step=<.050> min=<.100>`
Damping parameters for SCF iterations in order to reduce oscillations. The old Fock-operator is added to the current one with weight 0.5 as *start*; if convergence is good, this weight is then reduced by the *step* 0.05 in each successive iteration until the *min*imum of 0.1 is reached. (These are the default settings of define for closed-shell RHF). DSCF automatically tries to adjust the weight to optimize convergence but in difficult cases it is recommended to start with a large weight, e.g. 1.5, and to set the minimum to a larger value, e.g. 0.5.

`$scfdebug` *options*
Flags for debugging purposes. Following options are available:

`vectors` *integer*
Output level concerning molecular orbitals. *integer*=0 (default) means minimal output, >1 will output all start MOs and all MOs in each iteration.

`density` *integer*
Output level concerning difference density matrices.

`debug` *integer*
integer > 0 will dump a lot of information—be careful!

`$scfdenapproxl` *integer*
Direct SCF procedures build the Fock matrix by exploiting information from previous iterations for better efficiency. With this keyword information from the last *integer* iterations will be used. This feature is switched on with the default value 20, even if the keyword is absent. The user may reduce the number of iterations employed to smaller values (e. g. 10) in cases were numerical stability could become an issue. With the value 0 this feature is switched off; the Fock matrix is constructed from scratch in each iteration.

`$scfdiis` *options*
Control block for convergence acceleration via Pulay's DIIS *.
Options are:

*P.Pulay; Chem.Phys.Lett., **73**, 393 (1980), P.Pulay; J.Comput.Chem., **4**, 556 (1982)

`errvec=`*char* specifies the kind of error vector to be used (two different kind of DIIS algorithms)

char=`'FDS' or 'SDF' or 'FDS-SDF'`
uses (FDS − SDF) as error vector.

char= `none`
no DIIS

char= `sFDs`
use $S^{-1/2}FDS^{1/2}$ − transposed

Further suboptions:

`maxiter=`*integer*
maximum number of iterations used for extrapolation.

`debug=`*integer*
debug level (default: 0)

`integer=1`	print applied DIIS coefficients
`integer=2`	print DIIS matrix and eigenvalues, too

`qscal=`*real*
scaling factor in DIIS procedure: `qscal` > 1 implies some damping, `qscal` $= 1.0$: straight DIIS.

`thrd=`*real*
directs the reduction of qscal to `qscal` $= 1.0$ (no damping in DIIS), done if $||errvec|| <$ `thrd`.

Defaults for `$prediag` (see above) and `$scfdiis`

`errvec=FDS-SDF`, `maxiter=5`, `qscal=1.2`, `thrd=0.0`, this implies DIIS damping in all iterations, prediag is switched of.

Recommended:

`errvec=sFDs` leads to the following defaults:
`qscal=1.2`, for SCF runs: `maxiter=6` and `thrd=0.3`, prediag is off; for DFT runs: `maxiter=5` and `thrd=0.1` prediag is on. If you want to switch off prediag put
`$prediag none`.

`$scfdump`
Dump SCF restart information onto data group `$restartd` and dump SCF MOs in each iteration onto `$scfmo` (scfdump = iter). Additionally, a data block `$scfiterinfo` will be dumped containing accumulated SCF total-, one- and two-electron energies of all previous SCF iterations. Information that will allow you to perform a restart if your calculation aborts will be dumped on data group `$restartd` (see also `$restart`).

`$scfintunit` *options*
Disc space specification for two-electron integrals. The following suboptions are available (and necessary):

`unit=`*integer*
: Fortran unit number for this file. Unit numbers 30,31,... are recommended.

`size=`*integer*
: Filespace in megabytes for this file. `size=0` leads to a fully direct run. `size` is set by a statistics run, see `$statistics`. DSCF switches to direct mode if the file space is exhausted.

`file=`*char*
: Filename. This may also be a complete path name, if you want to store the integrals in a special directory. Make sure the file is local, otherwise integrals are transmitted over the network.

Thus your data group `$scfintunit` may look like this:

```
$scfintunit
 unit=30       size=35       file=twoint1
 unit=31       size=35       file=/users/work/twoint2
```

Maximal 30 files may be specified in this way.

`$scfiterlimit` *integer*
: Maximum number of SCF iterations (default: 30).

`$scfmo` *none* `file=`*char*
: Input/output data group for SCF MOs. You can specify

`none`
: To perform a calculation without a start vector (i.e. use a core Hamiltonian guess).

`file=`*char*
: The file where the MOs are written on output (default: mos).

These two options can also be used for `$uhfmo_alpha` and `$uhfmo_beta` to use a core guess and write the molecular orbitals to `file`.

After running `define` or a `TURBOMOLE` calculation additional options may appear specifying the origin of the MOs:

`expanded`
: These MOs were obtained by projection form another basis set. They should not be used for wavefunction analysis.

`scfconv=`*integer*
: The MOs are converged SCF MOs , the convergence criterion applied was $10^{-integer}$

`scfdump=`*integer*
: The MOs are unconverged SCF MOs which were written on this data group after iteration *integer*. The latter three options are mutually exclusive.

`format` *(format string)*
This specifies the FORTRAN format specification which was used for MO output. The standard format is (4d20.14). (See data group `$mo output format`.)

Example:
Your data group `$scfmo` could look like this after a successful `TURBOMOLE` run :

```
$scfmo  scfconv=7  format(3(1x,d19.13))
1  a1   eigenvalue=-.524127   nsao=6
.1234567890123d+01 -.1234567890123d+00  .1234567890123d-01
.1234567890123d+01 -.1234567890123d+00
3  a2   eigenvalue=-.234810
...
```

`$scforbitalorder on/off`
Order SCF MOs with respect to their energies (default: on)

`$scforbitalshift` *options*
To assist convergence, either the energies of unoccupied MOs can be shifted to higher energies or, in open-shell cases, the energies of closed-shell MOs to lower energies. In general a large shift may help to get better convergence.

Options are:

`noautomatic`
Automatic virtual shell shift switched off.

`automatic` *real*
Automatic virtual shell shift switched on; the energies of virtual orbitals will be shifted if the HOMO-LUMO gap drops below *real* such that a gap of *real* is sustained. This is the default setting if the keyword is missing with *real*=0.1.

`closedshell=`*real*
Option for open-shell cases. Closed shells are shifted to lower energies by *real*. The default shift value is `closedshell=0.4`.
Note: Normally this will disable the automatic shift of energies of virtual orbitals! To override this, you should append an exclamation mark to the 'automatic' switch, i.e. specify '`automatic!` *real*'.

`individual`
Set shifts for special occupied MOs. To use this option, start the line with the symmetry label and the list of MOs within this symmetry and append the desired shift in brackets as in the following example:

```
a1  1,2,4-6  (-.34)
b1  8        (+.3)
```

`$scftol` *real*

Integral evaluation threshold. Integrals smaller than *real* will not be evaluated. Note that this threshold may affect accuracy and the convergence properties if it is chosen too large. If `$scftol` is absent, a default value will be taken obtained from `$scfconv` by $real = \frac{10^{-(scfconv+1)}}{3 \cdot \#bf}$ (#bf = number of basisfunctions).

`$scratch files`

The scratch files allocated by `dscf` can be placed anywhere in your file systems instead of the working directory by referencing their pathnames in this data group. All possible scratch files are listed in the following example:

```
$scratch files
    dscf    dens          path1/file1
    dscf    fock          path2/file2
    dscf    dfock         path3/file3
    dscf    ddens         path4/file4
    dscf    statistics    path7/file7
    dscf    errvec        path8/file8
    dscf    oldfock       path9/file9
    dscf    oneint        path10/file10
```

The first column specifies the program type (`dscf` stands for SCF energy calculations, i. e. the `dscf` program), the second column the scratch file needed by this program and the third column the pathname of the file to be used as scratch file.

`$statistics` *options*

The following options are allowed

`off`	Do not perform integrals statistics
`dscf`	Perform integrals statistics for `dscf`
`kora`	see KORA
`mpgrad`	see `mpgrad`
`polly`	see POLLY
`dscf parallel`	see PARALLEL PROCESSING

Options `kora, dscf parallel, grad, mpgrad, polly` will be described in the related chapters.

If `$statistics dscf` has been given integral prescreening will be performed (which is an n^4-step and may therefore be time-consuming) and a table of the number of stored integrals as a function of the two parameters `$thize` and `$thime` will be dumped. Afterwards, the filespace needed for the current combination of `$thize` and `$thime` will be written to the data group (`$scfintunit`) and `$statistics dscf` will be replaced by `$statistics off`.

`$thime` *integer*

Integral storage parameter, which is related to the time needed to calculate the integral. The larger *integer* the less integrals will be stored. The default value is *integer* = 5. (see also `$thize`, `$statistics`)

`$thize` *real*

Integral storage parameter, that determines, together with `$thime`, the number of integrals stored on disc. Only integrals larger than *real* will be stored. The default value is *real* = 0.100E-04.

RHF/ROHF

`$closed shells`

Specification of MO occupation for RHF, e.g.

```
 a1g     1-4                                    ( 2 )
 a2g     1                                      ( 2 )
```

`$open shells type=1`

MO occupation of open shells and number of open shells. 'type=1' here means that there is only a single open shell consisting e.g. of two MOs:

```
b2g     1                                      ( 1 )
b3g     1                                      ( 1 )
```

`$rohf`

This data group is necessary for ROHF calculations with more than one open shell. Example:

```
$rohf          1
    a -a     a=0  b=0
    h -h     a=1  b=2
    a -h     a=1  b=2
```

This example is for the 7S state of chromium ($3d^5\,4s^1$) in symmetry group I. Note that for this option being activated, `$roothaan` also has to be specified in your `control` file, although its parameter has no meaning in this case. For more details see Section 6.3.

`$roothaan`

For ROHF-calculations with only one open shell the Roothaan parameters[†] a and b have to be specified within this data group (see also `$rohf`). Example:

```
$roothaan
    a = 3/4      b = 3/2
```

[†] C. C. J. Roothaan: Rev. Mod. Phys. **32** (1960) 179.

This example is for the 3P ground state of carbon ($2p^2$) in symmetry group I. `define` recognizes most cases and suggests good Roothaan parameters.

For further information on ROHF calculations (e.g. with more than one open shell), see the sample input in Section 15.6 and the tables of Roothaan parameters in Section 6.3.

Note that this keyword toggles the ROHF mode also for more than one open shell. If it is not given, the open-shell electrons are simply ignored.

UHF

`$alpha shells` and `$beta shells`

these two data groups specify the occupation of alpha and beta spin UHF MOs (syntax as any data group related with orbital occupation information, e.g. `$closed shells`)
Example:

```
$alpha shells
a      1-8                                    ( 1 )
b      1-2                                    ( 1 )
$beta shells
a      1-7                                    ( 1 )
b      1-3                                    ( 1 )
```

`$uhf`

directs the program to carry out a UHF run. `$uhf` overwrites closed-shell occupation specification.

`$uhfmo_alpha` and `$uhfmo_beta`

These two data groups contain the UHF MO vectors for alpha and beta spin respectively (same syntax as `$scfmo`).

`$uhfmo_beta`

see `$uhfmo_alpha`

DFT

```
$dft
 functional b-p
 gridsize m3
```

for DFT calculations one has to specify the functional and the grid (for the quadrature of the exchange correlation part). The settings above are default: both lines can be left out if the B-P86 functional and grid m3 are required. Other useful functionals supported are:

`b-lyp`

`b3-lyp`

`b3-lyp_Gaussian` (equivalent to the Gaussian98 keyword B3LYP with VWNIII)

`bh-lyp`

`s-vwn`

`s-vwn_Gaussian` (equivalent to the Gaussian98 keyword SVWN with VWNIII)

`tpss`

`tpssh`

Possible grids are 1–5 and m3–m5 where grid 1 is coarse (least accurate) and 5 most dense. We recommend however the use of so-called multiple grids m3–m5: SCF iterations with grid 1–3, final energy and gradient with grid 3–5. Usually m3 is fine: for large or delicate systems, try m4. For a reference calculation with a very fine grid and very tight thresholds use 'reference' as grid specification instead of 'gridsize xy'.

Note: the functionals b3-lyp_Gaussian and s-vwn_Gaussian are made available only for comparability with Gaussian. The functional VWNIII is much less well founded than VWN5 and the `TURBOMOLE` team does not recommend the use of VWNIII.

RI

`Dscf` does not run with the keyword $rij: you must call the RI modules `Ridft` and `Rdgrad` for energy and gradient calculations. However, it does run with the keyword $rik, but it will ignore all RI settings and do a conventional non-RI Hartree–Fock or DFT calculation.

`$rij`
: Enforces an RI-J calculation if module `ridft` is used, can be used for Hartree-Fock as well as for DFT calculations with pure or hybrid functionals.

`$ridft`
: Obsolete keyword - use $rij instead!

`$rik`
: Enforces a RI-JK calculation if module `ridft` is used, can be used for Hartree-Fock as well as for DFT calculations with pure or hybrid functionals.

`$ricore` *integer*
: Choose the memory core available (in megabyte) for special arrays in the RI calculation (the less memory you give the more integrals are treated directly, i.e. recomputed on the fly in every iteration)

`$jbas file=`*auxbasis*
: Cross reference for the file specifying the auxiliary basis as referenced in

$atoms. We strongly recommend using auxbasis sets optimized for the respective MO basis sets, e.g. use SVP (or TZVP) for the basis and the corresponding auxbasis as provided by define (default: file=auxbasis).

$ripop
Calculation of atomic charges according to the s partial wave and atomic dipole moments according to the p partial wave as resulting from the auxbasis representation of the density

RI-JK

If the keyword $rik is found in the control file, ridft performs a Hartree–Fock–SCF calculation using the RI-approximation for both Coulomb and HF-exchange (efficient for large basis sets). For this purpose needed (apart from $ricore):

$jkbas file=auxbasis
Cross reference for the file specifying the JK-auxiliary basis as referenced in $atoms. This group is created by the rijk menu in define.

MARI-*J*

Multipole-Accelerated-Resolution-of-Identity-J. This method partitions the Coulomb interactions in the near- and far-field parts. The calculation of the far-field part is performed by application of the multipole expansions and the near-field part is evaluated employing the RI-J approximation. It speeds up calculation of the Coulomb term for large systems. It can only be used with the ridft module and requires setting of the $ridft keyword.

```
$marij
  precision   1.0D-06
  lmaxmom     10
  nbinmax     8
  wsindex     0.0
  extmax      20.0
  thrmom      1.0D-18
```

The following options are available:

precision specifies precision parameter for the multipole expansions. Low-precision MARI-J calculations require $1 \cdot 10^{-6}$, which is the default. For higher precision calculations it should be set to $1 \cdot 10^{-8}$–$1 \cdot 10^{-9}$.

`lmaxmom` maximum l-moment of multipole expansions. It should be set to a value equal at least twice the maximum angular momentum of basis functions. Default value is 10 and it should probably never be set higher than 18.

`thrmom` Threshold for moment summation. For highly accurate calculations it should be set to $1 \cdot 10^{-24}$.

`nbinmax` number of bins per atom for partitioning of electron densities. Default value is 8 and hardly ever needs to be changed.

`wsindex` minimum separation between bins. Only bins separated more than the sum of their extents plus `wsindex` are considered as far-field. Default is 0.0 and should be changed only by the experts.

`extmax` maximum extent for charge distributions of partitioned densities. Extents with values larger then this are set to `extmax`. Hardly ever needs to be changed.

LHF

Use the Localized Hartree–Fock (LHF) method to obtain an effective Exact-Exchange Kohn–Sham potential (module `dscf`). The LHF method is a serial implementation for spin-restricted closed-shell and spin-unrestricted ground states.

```
$dft
 functional lhf
 gridsize 6
 test-integ
```

With the LHF potential Rydberg series of virtual orbitals can be obtained. To that end, diffuse orbital basis sets have to be used and special grids are required.

`gridtype 4` is the most diffuse with special radial scaling; `gridtype 5` is for very good Rydberg orbitals; `gridtype 6` (default in `Lhfprep`) is the least diffuse, only for the first Rydberg orbitals.

Only gridsize 3–5 can be used, no multiple grids.

`test-integ` will check if the selected grid is accurate enough for the employed basis-set by performing a numerical integration of the norm of all orbitals.

How to do LHF runs

1) Do a Hartree–Fock calculation using `dscf`.

2) Use the script `lhfprep` to prepare the `control` file (the old `control` file will be saved in `control.hf` and the molecular orbitals in `mos.hf` or in `alpha.hf` and `beta.hf` for the spin-unrestricted case). See `lhfprep -help` for options.

3) Run again `dscf`.

Otherwise the LHF functional can be selected in `define`: in this case default options are used.

Options for the LHF potential can be specified as follows (see also `lhfprep -help`)

```
$lhf
   off-diag    on
   numerical-slater off
   pot-file save
   asymptotic dynamic=1.d-3
   homo     1b1u
   homob    1b1u    # ONLY UNRESTRICTED
   conj-grad conv=1.d-7 maxit=20 output=1 cgasy=1
   slater-dtresh    1.d-9
   slater-region     7.0 0.5 10.0 0.5
   corrct-region             10.0 0.5
   slater-b-region   7.0 0.5 10.0 0.5 # ONLY UNRESTRICTED
   corrct-b-region           10.0 0.5 # ONLY UNRESTRICTED
   correlation func=lyp
```

`off-diag off`
: calculation of the KLI exchange potential. By default the LHF exchange potential is computed (`off-diag on`).

`numerical-slater on`
: the Slater potential is calculated numerically everywhere: this is more accurate but much more expensive.

`numerical-slater off`
: leads to accurate results only for first-row elements or if an uncontracted basis set or a basis set with special additional contractions is used: in other cases `numerical-slater on` has to be used (this is default).

`asymptotic`
: for `asymptotic` treatment there are three options:

  `asymptotic off`
  : No asymptotic-treatment and no use of the numerical Slater. The total exchange potential is just replaced by $-1/r$ in the asymptotic region. This method is the fastest one but can be used only for the density-matrix convergence or if Rydberg virtual orbitals are of no interest.

  `asymptotic on`
  : Full asymptotic-treatment and use of the numerical Slater in the near asymptotic-region.

`asymptotic dynamic=1.d-3`
: Automatic switching on (off) to the special asymptotic treatment if the differential density-matrix rms is below (above) 1.d-3. This is the default.

`pot-file save`
: the converged Slater and correction potentials for all grid points are saved in the files `slater.pot` and `corrct.pot`, respectively. Using `pot-file load`, the Slater potential is *not calculated* but read from `slater.pot` (the correction potential is instead recalculated). For spin unrestricted calculations the corresponding files are `slaterA.pot`, `slaterB.pot`, `corrctA.pot` and `correctB.pot`.

`homo`
: allows the user to specify which occupied orbital will not be included in the calculation of correction potential: by default the highest occupied orbital is selected. This option is useful for those systems where the HOMO of the starting orbitals (e.g. EHT, HF) is different from the final LHF HOMO. `homob` is for the beta spin.

`correlation func=`*functional*
: a correlation functional can be added to the LHF potential: use `func=lyp` for LYP, or `func=vwn` for VWN5 correlation.

For expert users

Options for the conjugate-gradient algorithm for the computation of the correction potential: rms-convergence (`conj-grad conv=1.d-7`), maxium number of iteration (`maxit=20`), output level `output=0-3`, asymptotic continuation in each iteration (`cgasy=1`).

With `slater-dtresh= 1.d-9` (default) the calculations of the numerical integrals for the Slater potential is performed only if it changes more than 1.d-9.

Asymptotic regions specification:

`corrct-region` R_F Δ_F
$0 \ldots R_F - \Delta_F$: basis-set correction potential
$R_F - \Delta_F \ldots R_F + \Delta_F$: smooth region
$R_F + \Delta_F \ldots + \infty$: asymptotic correction
Defaults: $R_F = 10$, $\Delta_F = 0.5$

`slater-region` R_N Δ_N R'_F Δ'_F
$0 \ldots R_N - \Delta_N$: basis-set Slater potential
$R_N - \Delta_N \ldots R_N + \Delta_N$: smoothing region
$R_N + \Delta_N \ldots R'_F - \Delta'_F$: numerical Slater
$R'_F - \Delta'_F \ldots R'_F + \Delta'_F$: smoothing region
$R'_F + \Delta'_F \ldots + \infty$: asymptotic Slater
Note: $R'_F - \Delta'_F \leq R_F - \Delta_F$
Defaults: $R_N = 7$, $\Delta_N = 0.5$, $R'_F = 10$, $\Delta'_F = 0.5$
Use `correct-b-region` and `slater-b-region` for the beta spin.

Two-component SCF (GHF)

Self-consistent two-component calculations (e.g. for spin-orbit interactions) can be carried out using the module `ridft` . The following keywords are valid:

`$soghf`
: enforces two-component-SCF calculations; this option is combinable with `$rij`, `$rik` and `$dft`.

`$kramers`
: switches on Kramers-restricted formalism

`$gdiis`
: enforces DIIS for complex Fock operator.

Scalar-relativistic Douglas–Kroll–Hess (DKH) Hamiltonian

Scalar-relativistic *all-electron* calculations can be done employing the Douglas–Kroll–Hess (DKH) Hamiltonian. Implemented for modules `dscf` and `ridft`.

`$dkhorder` *integer*
: switches on DKH calculation of order *integer*.

`$dkhparam` *integer*
: selects parametrization of the DKH Hamiltonian. Valid values are 1 (=default), 2, 3, 4, and 5.

14.2.6 Keywords for Periodic Electrostatic Embedded Cluster Method

The Periodic Electrostatic Embedded Cluster Method (PEECM) functionality provides electronic embedding of a finite, quantum mechanical cluster in a periodic, infinite array of point charges. It is implemented within HF and DFT energy and gradient `TURBOMOLE` modules: `dscf`, `grad`, `ridft`, `rdgrad`, and `escf`. Unlike embedding within a finite set of point charges the PEEC method always yields the correct electrostatic (Madelung) potential independent of the electrostatic moments of the point charges field. It is also significantly faster than the traditional finite point charges embedding.

The basic PEECM settings are defined in the `$embed` block. It can be redirected to an external file using `$embed file=<file_name>`.

Following keywords are used for the PEECM calculation setup:

`periodic`

Specifies the number of periodic directions. Allowed values for `number` are 3 for a bulk three-dimensional system, 2 for a two-dimenional surface slab, and 1 for a one-dimensional system. Default value is 3.

`cell`

Unit cell parameters in a form of six real values $|\mathbf{a}|$, $|\mathbf{b}|$, $|\mathbf{c}|$, α, β, γ, where $|\mathbf{a}|$, $|\mathbf{b}|$, $|\mathbf{c}|$ are lengths of the appropriate cell vectors, α is the angle between vectors $\mathbf{b}$ and $\mathbf{c}$, β is the angle between vectors $\mathbf{a}$ and $\mathbf{c}$, and γ is the angle between vectors $\mathbf{a}$ and $\mathbf{b}$. Default are atomic units and degrees. You can specify unit cell parameters in Å and degrees using `cell ang`.

```
content
  label x y z
end
```

Content of the unit cell, where `label` is the label of the point charge Content of the unit cell, where `label` is the label of the point charge type and `x y z` are corresponding Cartesian or fractional crystal coordinates. Defaults are Cartesian coordinates and atomic units. You can specify Cartesian coordinates in Å using `content ang` and fractional coordinates using `content frac`. Note that Cartesian coordinates assume that the cell vector $\mathbf{a}$ is aligned along the x axis, and the vector $\mathbf{b}$ on the xy plane.

```
cluster
  label x y z
end
```

Atomic coordinates of the piece of the crystal to be replaced by the QM cluster and surrounding isolation shell (ECPs amd explicit point charges), where `label` is the point charge label and `x y z` are corresponding Cartesian or fractional crystal coordinates. Defaults are Cartesian coordinates and atomic units. You can specify Cartesian coordinates in Å using `cluster ang` and fractional coordinates using `cluster frac`.

```
charges
  label charge
end
```

Values of point charges, where `label` is the point charge label and `charge` specifies charge in atomic units.

Additionally, the following keywords control the accuracy of PEECM calculation:

`lmaxmom`
: Maximum order of the multipole expansions in periodic fast multipole method (PFMM). Default value is 25.

`potval`
: Electrostatic potential at the lattice points resulting from periodic point charges field will be output if this keyword is present. Default is not to output.

`wsicl`
: Well-separateness criterion for PFMM. Default is 3.0.

`epsilon`
: Minimum accuracy for lattice sums in PFMM. Default is 1.0d-8.

14.2.7 Keywords for COSMO

The Conductor-like Screening Model (`cosmo`) is a continuum solvation model, where the solute molecule forms a cavity within the dielectric continuum of permittivity epsilon that represents the solvent. A brief description of the method is given in chapter 13. The model is currently implemented for SCF energy and gradient calculations (`dscf`/`ridft` and `grad`/`rdgrad`), MP2 energy calculations (RIMP2 and `mpgrad`) and MP2 gradients (RIMP2).

For simple HF or DFT single point calculations or optimizations with standard settings, we recommend to add the `$cosmo` keyword to the `control` file and to skip the rest of this section.

Please note: due to improvements in the **A** matrix and cavity setup the `cosmo` energies and gradients may differ from older versions (5.7 and older). The `use_old_amat` option can be used to calculate energies (not gradients) using the old cavity algorithm of TURBOMOLE 5.7.
The basic `cosmo` settings are defined in the `$cosmo` and the `$cosmo_atoms` block.
Example with default values:

```
$cosmo
  epsilon=infinity
  nppa= 1082
  nspa=   92
  disex= 10.0000
  rsolv= 1.30
  routf= 0.85
  cavity closed
  ampran= 0.1D-04
  phsran= 0.0
# the following options are not used by default
  allocate_nps= 140
  use_old_amat
  no_oc
```

`epsilon=`*real*
: defines a finite permittivity used for scaling the screening charges.

`allocate_nps=`*integer*
: skips the `cosmo` segment statistics run and allocates memory for the given number of segments.

`no_oc`
: skips the outlying charge correction.

All other parameters affect the generation of the surface and the construction of the **A** matrix:

`nppa=` *integer*
: number of basis grid points per atom
(allowed values: $k = 10 \times 3^i \times n^2 + 2 = 12, 32, 42, 92...$)

`nspa=` *integer*
: number of segments per atom
(allowed values: $k = 10 \times 3^i \times n^2 + 2 = 12, 32, 42, 92...$)

`disex=` *real*
: distance threshold for A matrix elements (Ångstrom)

`rsolv=` *real*
: distance to outer solvent sphere for cavity construction (Ångstrom)

`routf=` *real*
: factor for outer cavity construction in the outlying charge correction

`cavity closed`
: fill in seams between atoms with points

`cavity open`
: leave untidy seams between atoms

`ampran=` *real*
: amplitude of the cavity de-symmetrization

`phsran=` *real*
: phase of the cavity de-symmetrization

`use_old_amat`
: uses **A** matrix setup of `TURBOMOLE` 5.7

If the `$cosmo` keyword is given without further specifications the default parameter are used (recommended). For the generation of the cavity, `cosmo` also requires the definition of atomic radii. User defined values can be provided in Ångstrom units in the data group `$cosmo_atoms`, e.g. for a water molecule:

```
$cosmo_atoms
# radii in Angstrom units
```

```
o  1                                                    \
   radius=  1.7200
h  2-3                                                  \
   radius=  1.3000
```

If this section is missing in the `control` file, the default values defined in the `radii.cosmo` file (located in `$TURBODIR/parameter`) are used. A user defined value supersedes this defaults. `$cosmo` and `$cosmo_atoms` can be set interactively with the `cosmo` input program `Cosmoprep` after the usual generation of the `TURBOMOLE` input.

The `cosmo` energies and total charges are listed in the result section. E.g.:

```
  SCREENING CHARGE:
    cosmo      :  -0.003925
    correction :   0.003644
    total      :  -0.000282
  ENERGIES [a.u.]:
    Total energy            =      -76.0296831863
    Total energy + OC corr. =      -76.0297567835
    Dielectric energy       =       -0.0118029468
    Diel. energy + OC corr. =       -0.0118765440
    The following value is included for downward compatibility
    Total energy corrected  =      -76.0297199849
```

The dielectric energy of the system is already included in the total energy. `OC corr` denotes the outlying charge correction. The last energy entry gives the total outlying charge corrected energy in the old definition used in `TURBOMOLE` 5.7 and older versions.

Isodensity Cavity: This option can be used in HF/DFT single point calculations only. The `$cosmo_isodens` section defines the settings for the density based cavity setup (see also chapter 13). If the `$cosmo_isodens` keyword is given without suboptions, a scaled iosodensity cavity with default settings will be created. Possible options are:

`$cosmo_isodens`
: activates the density based cavity setup. The default values of `nspa` and `nsph` are changed to 162 and 92, respectively. This values are superseded by the user defined `nspa` value of the `$cosmo` section. By default the scaled density method is used. The atom type dependent density values are read from the `radii.cosmo` file (located in `$TURBODIR/parameter`).

`dx=`*real*
: spacing of the marching tetrahedron grid in Å (defualt: 0.3Å).

`all_dens=`*real*
: use one isodensity value for all atom types (value in a.u.)

The outlying charge correction will be performed with a radii based outer cavity. Therfore, and for the smoothing of the density changes in the intersection areas of the scaled density method, radii are needed as for the standard COSMO cavity. **Please note:** The isodensity cavity will be contructed only once at the beginnig of the SCF calculation. The density contructed from the initial mos will be used (file `mos` or `alpha/beta` in case of unrestricted calculations). Because the mos of an initial guess do not provide a good density for the cavity construction, it is necessary to provide mos of a converged SCF calculation (e.g. a COSMO calculation with standard cavity). We recommend the following three steps: perform a standard COSMO calculation, add the isodensity options afterwards, and start the calculation a second time.

COSMO **in MP2 Calculations:** The iterative `cosmo` PTED scheme (see chapter 13) can be used with the `mp2cosmo` script. Options are explained in the help message (`mp2cosmo -h`). Both MP2 modules RIMP2 and `mpgrad` can be utilized. The `control` file can be prepared by a normal `cosmo` SCF input followed by a RIMP2 or `mpgrad` input. The PTE gradients can be switched on by using the

```
$cosmo_correlated
```

keyword (RIMP2 only). Again a normal SCF `cosmo` input followed by a RIMP2 input has to be generated. The `$cosmo_correlated` keyword forces `dscf` to keep the `cosmo` information needed for the following MP2 calculation and toggles on the `cosmo` gradient contribution.

COSMO **in Numerical Frequency Calculations:** `NumForce` can handle two types of `cosmo` frequency calculations. The first used the normal relaxed `cosmo` energy and gradient. It can be performed with a standard `dscf` or `ridft` COSMO input without further settings. This is the right method to calculate a Hessian for optimizations. The second type, which uses the approach described in chapter 13, is implemented for `ridft` only. The input is the same as in the first case but `Numforce` has to be called with the `-cosmo` option. If no solvent refractive index `refind=`REAL is given in the `$cosmo` section of the control file the program uses the default (1.3).

14.2.8 Keywords for Modules GRAD and RDGRAD

Many of the `dscf` and `ridft` keywords are also used by `grad` and `rdgrad`.

```
$drvopt
```

This keyword and corresponding options are required in gradient calculations only in special circumstances. Just `$drvopt` is fine, no options needed to compute derivatives of the energy with respect to nuclear coordinates within the method specified: SCF, DFT, RIDFT.

If running a DFT gradient calculation, it is possible to include the derivatives of the quadrature weights, to get more accurate results. In normal cases however those effects are marginal. An exception is numerical calculation of frequencies by `Numforce`, where it is strongly recommended to use the weight derivatives option. The biggest deviations from the uncorrected results are to be expected if doing gradient calculations for elements heavier than Kr using all electron basis sets and very small grids. To use the weight derivatives option, add

```
weight derivatives
```

in `$dft`.

The option

```
point charges
```

in `$drvopt` switches on the evaluation of derivatives with respect to coordinates of point charges. The gradients are written to the file `$point_charge_gradients` old gradients will be overwritten.

14.2.9 Keywords for Module AOFORCE

This module calculates analytically harmonic vibrational frequencies within the HF- or (RI)DFT-methods for closed-shell and spin-unrestricted open-shell-systems. Broken occupation numbers would lead to results without any physical meaning. Note, that RI is only used partially, which means that the resulting Hessian is only a (very good) approximation to exact second derivatives of the RIDFT-energy expression. Apart from a standard force constant calculation which predicts all (allowed and forbidden) vibrational transitions, it is also possible to specify certain irreps for which the calculation has to be done exclusively or to select only a small number of lowest eigenvalues (and eigenvectors) that are generated at reduced computational cost.

General keywords

`$drvopt`
: is the keyword for non-default options of gradient and second derivative calculations. Possibilities in case of the module `aoforce` are:

  `frequency analysis only`

  `analysis only`
  : to read a complete Hessian from the input file `$hessian` and perform only the frequency analysis

  `analysis [only] intcoord [print` *printlevel*`]`
  : to perform an analysis of normal modes in terms of internal coordinates.

Details about this option and the effect of the printlevel (default is 0) are given in Section 10. The effect of the keyword `only` is the same as described above.

`$maxcor 50`

fixes the RAM memory to be used by the run (here 50 MB), about 70% of available memory should be fine, because `$maxcor` specifies only the memory used to store derivatives of density and Fock matrices as well as the CPHF-RHS. Default is 200 MB.

`$forceconv 7`

sets the convergence criterion for the CPHF-equations to a residual norm of 1.0d-7. Normally the default value of 1.0d-5 already provides an accuracy of vibrational frequencies of $0.01\,\mathrm{cm}^{-1}$ with respect to the values obtained for the convergence limit.

`$forceiterlimit 10`

fixes the maximum number of Davidson iterations for the solution of the CPHF-equations to a value of ten. Normal calculations should not need more than eight iterations, but as a precaution the default value is 25.

`$nosalc`

forces the program in case of molecules with C_1 symmetry not to use $3N-6(5)$ symmetry adapted but all $3N$ cartesian nuclear displacement vectors. This option may lead to a moderate speed-up for molecules notedly larger than 1000 basis functions and 100 atoms.

`$noproj`

forces the program not to project out translations and rotations when forming a basis of symmetry adapted molecular displacements. This option may be needed if a Hessian is required, that contains translation- and rotation-contributions, e.g. for coupling the system with low cost methods. Output of the unprojected hessian is done on `$nprhessian`; format is the same as for conventional `$hessian`. Output of the corresponding eigenvalues and eigenvectors is done analogously on `$nprvibrational spectrum` and `$nprvibrational normal modes`.

`$nomw`

causes the program to diagonalize a not mass weighted hessian. Output is on `$nprhessian`, `$nprvibrational spectrum` and `$nprvibrational normal modes`, because projection of rotations is not possible in this case.

`$isosub`

This keyword allows to trace back the effects of isotopic substitution on vibrational frequencies. The atom(s) for which isotopic substitution is to be investigated are specified in subsequent lines of the form (atom index) (mass in special isotope), e.g.

```
$isosub
 3 2.001
 5 13
```

The interpolation then takes place between the mass(es) specified in `$atoms` (or the default mass(es), if none specified) and the mass(es) in `$isosub`. Take care of symmetry equivalent atoms, otherwise symmetry analysis will fail. This feature can not be used in a lowest eigenvalue search (keyword `$les`).

`$isopts 6`

Sets the number of points for interpolation between the two isotopes compared by the `$isosub` option to six. Default value is 21.

Keywords for the treatment of only selected nuclear displacement vectors:

`$ironly`

CPHF-iteration is done only for distortions, that are IR active.

`$ramanonly`

CPHF-iteration is done only for distortions, that are Raman active.

`$les`

This causes a lowest Hessian eigenvalue search to be performed instead of a complete force constant calculation. The lowest eigenvalue search consists of the calculation of a guess-Hessian and macro-iterations to find the solution vector(s) for the lowest eigenvalue(s). In each macro-iteration the CPHF-equations are solved for the present search vector(s). `$les all 1` means that one lowest eigenvalue for each irrep will be determined, other numbers of lowest eigenvalues per irrep are admissible too.

Different numbers of lowest eigenvalues for different irreps are requested by e.g.

```
$les
 a1 3
 a2 all
 b2 1
```

The convergence criterion of the Davidson iterations for the solution of the CPHF-equations as well as the maximal residual norm for the lowest Hessian eigenvalue in the macro-iteration are specified by `$forceconv` as explained above.

The maximum number of macro-iterations is specified by `$lesiterlimit x`

with the default `x`=25. The maximum number of iterations for each solution of the CPHF-equations is again determined by `$forceiterlimit` as shown above.

The convergence of the macro-iterations is strongly influenced by the size of the starting search-subspace. Generally all guess-Hessian eigenvectors corresponding to imaginary frequencies and at least two real ones are used as

starting search-subspace. However it proved to be necessary to use even more vectors in the case of guess-Hessians with very large conditioning numbers.
`$hesscond 8.0d-5`
means that all eigenvalues with the quotient (eigenvalue)/(max. eigenvalue) lower than 0.00008 are added to the starting search-subspace. Default is `1.0d-4`.

Force constant calculations on the DFT level prove to be numerically reliable only with large integration grids or if one includes the effects of quadrature weights. This is done by default—to prevent this, insert

```
no weight derivatives
```

in `$dft`.

14.2.10 Keywords for Module ESCF

ESCF **calculations**

to perform an `escf` calculation converged molecular orbitals from a HF, DFT or RIDFT calculation are needed. The HF, DFT or RIDFT method is chosen according to the `$dft` or `$ridft` keywords, specified above. It is recommended to use well-converged orbitals, specifying `$scfconv 7` and `$denconv 1d-7` for the ground-state calculation. The input for an `escf` calculation can be conveniently generated using the `ex` menu in `define`, see Section 4.

In an `escf` run one of the following properties can be calculated: (please note the 'or' in the text, do only one thing at a time.)

1. RPA and time-dependent DFT singlet or triplet or spin-unrestricted excitation energies (HF+RI(DFT))

`$scfinstab rpas` or

`$scfinstab rpat` or

`$scfinstab urpa`

2. CI singles singlet or triplet or spin-unrestricted excitation energies (HF)

`$scfinstab ciss` or

`$scfinstab cist` or

`$scfinstab ucis`

3. Eigenvalues of singlet or triplet or non-real stability matrices (HF+RI(DFT), RHF)

`$scfinstab singlet` or

`$scfinstab triplet` or

`$scfinstab non-real`

4. Static polarizability and rotatory dispersion tensors (HF+(RI)DFT, RHF+UHF)

`$scfinstab polly`

5. Dynamic polarizability and rotatory dispersion tensors (HF+(RI)DFT, RHF+UHF)

`$scfinstab dynpol` *unit*
list of frequencies

where *unit* can be eV, nm, rcm; default is a.u. (Hartree). For example, to calculate dynamic polarizabilities at 590 nm and 400 i nm (i is the imaginary unit):

```
$scfinstab dynpol nm
  590
  400 i
```

The number and symmetry labels of the excited states to be calculated is controlled by the data group `$soes`. Example:

```
$soes
b1g  17
eu   23
t2g  all
```

will yield the 17 lowest excitations in IRREP b1g, the 23 lowest excitations in IRREP eu, and all excitations in IRREP t2g. Specify `$soes all` textitn; to calculate the n first excitations in all IRREPS. If n is not specified, all excitations in all IRREPS will be obtained.

During an `escf` run, a system-independent formatted logfile will be constructed for each IRREP. It can be re-used in subsequent calculations (restart or extension of eigenspace or of `$rpaconv`). An `escf` run can be interrupted by typing "touch stop" in the working directory.

general keywords

`$rpacor` n

The maximum amount of core memory to be allocated for the storage of trial vectors is restricted to n MB. If the memory needed exceeds the threshold

given by `$rpacor`, a multiple pass algorithm will be used. However, especially for large cases, this will increase computation time significantly. The default is 200 MB.

`$spectrum` *unit*
: The calculated excitation energies and corresponding oscillator strengths are appended to a file named 'spectrum'. Possible values of `unit` are eV, nm and cm^{-1} or rcm. If no unit is specified, excitation energies are given in a.u.

`$cdspectrum` *unit*
: The calculated excitation energies and corresponding rotatory strengths are appended to a file named 'cdspectrum'. `unit` can have the same values as in `$spectrum`.

`$start vector generation` *e*
: Flag for generation of UHF start MOs in a triplet instability calculation. The option will become effective only if there are triplet instabilities in the totally symmetric IRREP. The optional real number e specifies the approximate second order energy change in a.u. (default: 0.1).

`$velocity gauge`
: Enables calculation of dipole polarizability/rotatory dispersion in the velocity gauge. Active only for pure DFT (no HF exchange).

`$sum rules` *unit*
list of frequencies
: Enable calculation of oscillator and rotatory strength sum rules at frequencies specified by *list of frequencies* in unit *unit* (see `$scfinstab dynpol`). Note that the sums will be taken only over the states specified in `$soes`.

`$rpaconv` n
: the vectors are considered as converged if the Euclidean residual norm is less than 10^{-n}. Larger values of n lead to higher accuracy. The default is a residual norm less than 10^{-5}.

`$escfiterlimit` n
: Sets the upper limit for the number of Davidson Iterations to n. Default is $n = 25$.

14.2.11 Keywords for Module EGRAD

`egrad` uses the same general keywords as `escf` and `grad`, see Sections 14.2.8 and 14.2.10.

The state to be optimized is by default the highest excited state specified in `$soes`. Note that only one IRREP can be treated at the same time in contrast to `escf` calculations. When the desired excited state is nearly degenerate with another state of the same symmetry it may be necessary to include higher states in the initial

calculation of the excitation energy and vector in order to avoid root flipping. This is accomplished by means of the additional keyword

`$exopt` n

which explicitly enforces that n-th excited state is optimized. n must not be larger than the number of states specified in `$soes`.

14.2.12 Keywords for Modules MPGRAD and RIMP2

If an MP2 run is to be performed after the SCF run, the SCF run has to be done with at least

1) density convergence `$denconv 1.d-7`
2) energy convergence `$scfconv 6`

Keywords Valid for Both MPGRAD and RIMP2

`$maxcor` n

The data group `$maxcor` adjusts the maximum size of core memory (n in MB) which will be allocated during the MP2 run. Recommendation: 3/4 of the actual main memory at most. If `$maxcor` is not found, its value is set to 200 MB.

`$mp2energy`

Calculation of MP2 gradient is omitted, only MP2 energy is calculated. In connection with this keyword you may also activate the spin-component-scaled (SCS) MP2 proposed by Grimme

`$mp2energy SCS`

with the default values of 6/5 for pS and 1/3 for pT, which may be modified this way:

`$mp2energy SCS pt=`$val1$ `ps=`$val2$

```
$freeze
 a1g     1-2
 t1u     1
```

The data group `$freeze` specifies frozen orbitals, in the above syntax by irreducible representations. The symmetry-independent and for standard-applications recommended syntax is

```
$freeze
 implicit core=5 virt=2
```

This will freeze the 5 lowest occupied and 2 highest virtual orbitals (alpha and beta count as one in UHF cases). Note that degenerate orbitals count twice (e representations), thrice (t representations) etc. In case of `mpgrad` frozen orbitals have to be specified manually, for `rimp2` the preparation tool `rimp2prep` may be used to specify frozen core orbitals, frozen virtuals have to be specified manually. Note: In case of gradient calculations frozen core orbitals are regarded only by `rimp2`, but not by `mpgrad`, moreover freezing of virtual orbitals is generally not supported by `mpgrad`.

MPGRAD: Essential Keywords

All essential data groups for `mpgrad` may be generated by the preparation tool `mp2prep`, apart from `$maxcor` (see above) these are the following:

`$traloop` n
: specifies the number of loops (or 'passes') over occupied orbitals, n, performed in the `mpgrad` run: the more passes the smaller file space requirements—but CPU time will go up.

```
$mointunit
 type=intermed unit=61       size=0        file=halfint
 type=1111     unit=62       size=0        file=moint#0
 type=1112     unit=63       size=0        file=moint#1
 type=1122     unit=64       size=0        file=moint#j
 type=1212     unit=65       size=0        file=moint#k
 type=1212a    unit=70       size=0        file=moint#a
 type=gamma#1  unit=71       size=0        file=gamma#1
 type=gamma#2  unit=72       size=0        file=gamma#2
 type=1212u    unit=73       size=0        file=moint#u
 type=1112u    unit=74       size=0        file=moint#v
 type=gamma#1u unit=75       size=0        file=gamma#1u
```

The data group `$mointunit` specifies:

- which scratch files are needed,
- where they are located (path name) and
- (after a statistics run, see below) an estimated file size.

`$statistics mpgrad`
: statistics run (estimation of disc space needed) for the adjustment of the file sizes will be performed.

MPGRAD: Optional Keywords

`$mp2pair`
: calculation of MP2 pair correlation energies.

RIMP2: Essential Keywords

Apart from keywords `$maxcor`, `$mp2energy` and `$freeze` (see above) `rimp2` also needs

`$cbas file=auxbasis`
: cross reference for the file specifying the auxiliary basis as referenced in `$atoms`. We strongly recommend using auxbasis sets optimized for the corresponding MO basis sets.

Reasonable settings for these keywords may be generated by the tool `Rimp2prep`. Moreover you may specify by hand:

`$tmpdir /work/thisjob`
: specification of directory for scratch files; by default files are written to the working directory; works also with capital letters (for consistency with ricc2).

`$c1algorithm`
: avoids symmetry gymnastics in case of C_1-symmetry, rather for debugging

`$cbasopt`
: enforces calculation of

$$\frac{-|<ij||ab>_{(exact)} - <ij||ab>_{(RI)}|^2}{(\epsilon(i)+\epsilon(j)-\epsilon(a)-\epsilon(b))},$$

necessary for characterisation of auxiliary basis set quality and for auxiliary basis optimizations; works only for C_1-symmetry.
Note: all integrals are kept in memory, so this is for atoms and small molecules only.

`$tplot`
: Enforces plotting of five largest t-amplitudes as well es five largest norms of t-amplitudes for fixed pair of occupied orbitals ij. By additional integer this number may be changed.

`$mp2occ`
: Enforces plotting of all eigenvalues of the MP2 density matrix.

14.2.13 Keywords for Module RICC2

Note that beside the keywords listed below the outcome of the `ricc2` program also depends on the settings of most thresholds that influence the integral screening (e.g. `$denconv`, `$scfconv`, `$scftol`) and for the solution of Z vector equation with 4-index integrals (for relaxed properties and gradients) on the settings for integrals storage in semi-direct SCF runs (i.e. `$thime`, `$thize`, `$scfintunit`). For the explanation of these keywords see Section 14.2.5.

`$cbas file=auxbasis`
: Auxiliary basis set for RI approximation. For details Section 14.2.12.

`$freeze`
: Freeze orbitals in the calculation of correlation and excitation energies. For details see Section 14.2.12.

`$printlevel` 1
: Print level. The default value is 1.

`$tmpdir /work/thisjob`
: Specify a directory for large intermediate files (typically three-index coulomb integrals and similar intermediates), which is different from the directory where the `ricc2` program is started.

`$maxcor` 20
: The data group `$maxcor` adjusts the maximum size of core memory in MB which will be allocated during the RI-CC2 run. This keyword can be set in `define` or with the `Rimp2prep` tool, the default is 20 MB.
`$maxcor` has a large influence on computation times for RI-CC2 runs! It is recommended to set `$maxcor` to ca. 75–85% of the available (physical) core memory.

`$spectrum` *unit*
: The calculated excitation energies and corresponding oscillator strengths are appended to a file named 'spectrum'. Possible values of `unit` are eV, nm and cm^{-1} or rcm. If no unit is specified, excitation energies are given in a.u.

`$cdspectrum` *unit*
: The calculated excitation energies and corresponding rotatory strengths are appended to a file named 'cdspectrum'. `unit` can have the same values as in `$spectrum`.

`$laplace`

```
conv = 5
```

: The purpose of this data group is twofold: It activates the Laplace-transformed implementation of SOS-MP2 in the `ricc2` module (if the `sos` option has been specified in `$ricc2`) and it provides the options to specify the technical details for the numerical Laplace-transformation.

conv
: Threshold for the numerical integration used for the Laplace transformation of orbital energy denominators. The grid points for the numerical integration are determined such that is the remaining root mean squared error (RMSE) of the Laplace transformation is $< 10^{-\mathrm{conv}}$. By default the threshold is set to the value of `conv` given in `$ricc2` (see below).

`$ricc2`

```
ccs
cis
mp2     d1diag
cis(d)  energy only
cis(dinf)
adc(2)
cc2
restart
norestart
hard_restart
nohard_restart
conv    = 8
oconv   = 7
lindep  = 15
maxiter = 25
mxdiis  = 10
maxred  = 100
iprint  = 1
fmtprop = f15.8
geoopt model=cc2 state=(a" 2)
scs  cos=1.2d0   css=0.3333d0
sos
gsonly
d1diag
```

specifies the *ab initio* models (methods) for ground and excited states and the most important parameters and thresholds for the solution of the cluster equations, linear response equations or eigenvalue problems. If more than one model is given, the corresponding calculations are performed successively. Note: The CCS ground state energy is identical with the SCF reference energy, CCS excitation energies are identical to CIS excitation energies. The MP2 results is equivalent to the result from the `rimp2` module. `cis(dinf)` denotes the iterative CIS(D) variant CIS(D_∞).

`mp2 d1diag`
: Request the calculation of the D_1 diagnostic in MP2 energy calculations (ignored in MP2 gradient calculations). Note that the evaluation of the D_1 diagnostic increases the computational costs of the RI-MP2 energy calculation roughly by a factor of 3.

`cis(d) energy only`
: If the `energy only` flag is given after the `cis(d)` keyword, it is assumed that only excitation energies are requested. This switches on some shortcuts to avoid the computation of intermediates needed e.g. for the generation of improved start vectors for CC2.

`(no)restart`
: If the `restart` flag is set, the program will try to restart the CC2 calculations from previous solution vectors on file. If the `norestart` flag is set no restart will be done. Default is to do a restart for CC2 if and only if the file `CCR0--1--1---0` exists. **Note**: There is no restart possibility for CCS/CIS or MP2/CIS(D).

`(no)hard_restart`
: If the `hard_restart` flag is set, the program will try to reuse integrals and intermediates from a previous calculation. This requires that the `restart.cc` file has been kept, which contains check sums and some other informations needed. The `hard_restart` flag is switched on by default, if the `restart.cc` file is present.

`conv` The `conv` parameter gives the convergence threshold for the CC2 ground state energy as $10^{-\mathrm{conv}}$. The default value is taken from the data group `$deneps`.

`oconv`
: The `oconv` parameter gives an additional threshold for the residual of the cluster equations (vector function). If this parameter is given, the iterations for the cluster equations are not stopped before the norm of the residual is $< 10^{-\mathrm{oconv}}$. By default the threshold is set to `oconv` =`conv`-1, or $10\times$ `deneps` if no input for `conv` is given.

`lindep`
: If the norm of a vector is smaller than $10^{-\mathrm{lindep}}$, the vector is assumed to be zero. This threshold is also used to test if a set of vectors is linear dependent. The default threshold is 10^{-15}.

`maxiter`
: gives the maximum number of iterations for the solution of the cluster equations, eigenvalue problems or response equations (default: 25).

`mxdiis`
: is the maximum number of vectors used in the DIIS procedures for CC2 ground state or excitation energies (default: 10).

`maxred`
: the maximum dimension of the reduced space in the solution of linear equations (default: 100).

`iprint`
: print level, by default set to 1 or (if given) the the value of the `$printlevel` data group.

`fmtprop`
: Fortran print format used to print several results (in particular one-electron properties and transition moments) to standard output.

`geoopt`
: specify wavefunction and electronic state for which a geometry optimization is intended. For this model the gradient will be calculated and the energy and gradient will be written onto the data groups `$energy` and `$grad`. Required for geometry optimizations using the `jobex` script. Note, that in the present version gradients are only available for ground states at the MP2 and CC2 and for excited states at the CC2 level and not for ROHF based open-shell calculations. Not set by default. The default model is CC2, the default electronic state the ground state. To obtain gradients for the lowest excited state (of those included in the excitation energy calculation, but else of *arbitrary* multiplicity and symmetry) the short cut `s1` can be used. `x` is treated as synonym for the ground state.

`scs`
: the opposite–spin scaling factor `cos` and the same–spin scaling factor `css` can be choosen. If `scs` is set without further input, the SCS parameters cos=6/5 and css=1/3 are applied. This keyword can presently only be used in connection with MP2.

`sos`
: the SOS parameters cos=1.3 and css=0.0 are applied. This keyword can presently only be used in connection with MP2.

`d1diag`
: request the calculation of the D_1 diagnostic for the ground state wavefunction. Only needed for MP2 (see above for the alternative input option `mp2 d1diag`). For all other correlated methods the D_1 diagnostic is evaluated by default (without significant extra costs).

`$rir12`

`ansatz`

`r12model`

`comaprox`

`cabs`

`examp`

`pairenergy`

`local`

`corrfac`

`cabsingles`

`ump2fixed`

`ansatz` *char*

char=`1`, `2*` or `2`

The `ansatz` flag determines which ansatz is used to calculate the RI-MP2-F12 ground state energy.

(Ansatz 2 is used if `ansatz` is absent.)

`r12model` *char*

char=`A`, `A'` or `B`

The `r12model` flag determines which approximation model is used to calculate the RI-MP2-F12 ground state energy.

(Ansatz B is used if `r12model` is absent.)

`comaprox` *char*

char=`F+K` or `T+V`

The `comaprox` flag determines the method used to approximate the commutator integrals $[T, f_{12}]$.

(Approximation `T+V` is used if `comaprox` is absent.)

`cabs` *char val*

char=`svd` or `cho`

The `cabs` flag determines the method used to orthogonalize the orbials of the CABS basis. *val* is the threshold below which CABS orbitals are removed from the calculation.

(`svd 1.0d-08` is used if `cabs` is absent.)

`examp` *char*

char=`noinv`, `fixed` or `inv`

The `examp` flag determines which methods are used to determine the F12 amplitudes. For `inv` the amplitudes are optimised using the orbital-invariant method. For `fixed` and `noinv` only the diagonal amplitudes are non-zero and are either predetermined using the coalescence conditions (`fixed`), or optimised (`noinv`—not orbital invariant). If *char*=`inv`, the F12 energy contribution is computed using all three methods.

(The `fixed` method is used if `examp` is absent.)

`pairenergy` *char*

char=`off` or `on`

If *char*=`off` (default), the print out of the F12 contribution to the pair energies is supressed. The summary of the RI-MP2-F12 correlation energies is always printed out.

`local` *char*

char=`off`, `boys` or `pipek`

The active occupied molecular orbitals are localized by Boys or Pipek-Mezey method. Currently, the `local` flag is restricted to closed shell cases within approximation A and the linear correlation factor. If `local` is absent, no localization is performed.

`corrfac` *char*
: *char*=`LCG` or `R12`
The `corrfac` flag determines which correlation factor is used for the geminal basis. `LCG` requires the data group `$lcg`, which contains the information regarding exponents and coefficients of the linear combination of Gaussians.

`cabsingles` *char*
: *char*=`off` or `on`
The `cabsingles` flag determines whether or not the single excitations into the CABS basis are computed.
The CABS singles are computed in any case if the CABS Fock matrix elements are computed anyway for the F12 calculation (*i.e.*, for ansatz `2` or r12model `B` or comaprox `F+K`).

`ump2fixed` *char*
: *char*=`diag` or `full`
The `umpfixed` flag controls which fixed-amplitude method is used for calculations using ROHF or UHF references. `full` is more computationally demanding than `diag`, but gives energies closer to the `inv` method. If `ump2fixed` is absent, the `full` method is used.

```
$excitations

irrep=au  multiplicity=1 nexc=4  npre=6  nstart=8
irrep=bg  multiplicity=3 nexc=2  npre=4  nstart=5
spectrum states=all operators=diplen,dipvel
exprop   states=all operators=qudlen
xgrad    states=(ag{3} 1)
conv    = 6
thrdiis = 2
preopt  = 3
leftopt
bothsides
```

In this data group you have to give additional input for calculations on excited states:

`irrep`
: the irreducible representation.

`multiplicity`
: spin multiplicity (1 for singlet, 3 for triplet); default: singlet, not needed for UHF.

`nexc` the number of excited states to be calculated within this irrep and for this multiplicity.

`npre` the number of roots used in preoptimization steps (default: `npre` = `nexc`).

`nstart`
the number of start vectors generated or read from file (default: `nstart` = `npre`).

`spectrum`
This flag switches on the calculation of oscillator strengths for excited state—ground state transitions. Setting the parameter `states=all` is mandatory for the calculation of transition properties in the present version. The `operators` flag can be followed by a list of operators (see below) for which the transition properties will be calculated. Default is to compute the oscillator strengths for all components of the dipole operator.

`exprop`
require calculation of first-order properties for excited states. For the `states` option see `spectrum` option above; for details for the `operators` input see below.

`xgrad`
request calculation of the gradient for the total energy of an excited state. If no state is specified, the gradient will be calculated for the lowest excited state included in the calculation of excitation energies (Note that only a single state should be specified; simultaneous calculation of gradients for several states is in the present version not possible.).

`conv` convergence threshold for norm of residual vectors in eigenvalue problems is set to $10^{-\texttt{conv}}$. If not given, a default value is used, which is chosen as $\max(10^{-\texttt{conv}}, 10^{-\texttt{oconv}}, 10^{-6})$, where `conv` refers to the values given in the data group `$ricc2`.

`preopt`
convergence threshold used for preoptimization of CC2 eigenvectors is set to $10^{-\texttt{preopt}}$ (default: 3).

`thrdiis`
threshold ($10^{-\texttt{thrdiis}}$) for residual norm below which DIIS extrapolation is switched on in the modified Davidson algorithm for the non-linear CC2 eigenvalue problem (default: 2).

`leftopt`
If the flag `leftopt` is set the left eigenvectors are computed (default is to compute the right eigenvectors, for test purposes only).

`bothsides`
The `bothsides` flag enforces the calculation of both, the left and the right eigenvectors (for test purposes only).

`$response`

```
fop unrelaxed_only operators=diplen
```

```
gradient
conv = 6
zconv = 6
semicano
nosemicano
thrsemi = 3
```

In this data group you have to give additional input for the calculation of ground state properties and the solution of response equations:

`fop` This flag switches on the calculation of ground state first-order properties (expectation values). The `operators` flag can be followed by a list of operators (see below) for which the first-order properties will be calculated. Default is to compute the components of the dipole and the quadrupole moment. The option `unrelaxed_only` suppress the calculation of orbital-relaxed first-order properties, which require solution the CPHF-like Z-vector equations. Default is the calculation of unrelaxed and orbital-relaxed first-order properties. The `unrelaxed_only` option will be ignored, if the calculation of gradients is requested (see `gradient` option below and `geoopt` in data group `$ricc2`).

`gradient`
require calculation of geometric gradients. In difference to the `geoopt` keyword in the data group `$ricc2` this can be used to compute gradients for several methods within a loop over models; but gradients and energies will not be written to the data groups `$grad` and `$energy` as needed for geometry optimizations. Note, that in the present version gradients are only available for MP2 and CC2 and only for a closed-shell RHF reference.

`conv` convergence threshold for norm of residual vectors in linear response equations is set to $10^{-\texttt{conv}}$. If not given in the `$response` data group, a default value is used, which is chosen as $\max(10^{-\texttt{conv}}$, $10^{-\texttt{oconv}}, 10^{-6})$, where `conv` and `oconv` refer to the values given in the data group `$ricc2`.

`zconv`
convergence threshold for the norm of the residual vector in the solution of the Z vector equations will be set to $10^{-\texttt{zconv}}$.

`semicano`
use semi-canonical formulation for the calculation of (transition) one-electron densities. Switched on by default. The semi-canonical formulation is usually computationally more efficient than the non-canonical formulation. Exceptions are systems with many nearly degenerate pairs of occupied orbitals, which have to be treated in a non-canonical way anyway. (See also explanation for `thrsemi` below).

`nosemicano`
: use non-canonical formulation for the calculation of (transition) one-electron densities. Default is to use the semi-canonical formulation.

`thrsemi`
: the threshold for the selection of nearly degenerate pairs of occupied orbitals which (if contributing to the density) have to be treated in a non-canonical fashion will be set to $10^{-\texttt{thrsemi}}$. If set to tight the semi-canonical algorithm will become inefficient, if the threshold is to large the algorithm will become numerical unstable

`zpreopt`
: threshold for preoptimizating the so-called Z vector (i.e. the lagrangian multipliers for orbital coefficients) with a preceding RI-CPHF calculation with the cbas auxiliary basis. The RI-CPHF equations will be converged to a residual error $< 10^{-\texttt{zpreopt}}$. Default is `zpreopt=4`. This preoptimization can reduce significantly the computational costs for the solution of the Z vector equations for large basis sets, in particular if they contain diffuse basis functions. For calculations on large molecules with small or medium sized basis sets the preoptimization becomes inefficient compared to the large effects of integral screening for the conventional CPHF equations and should be disabled. This option is automatically disabled for `ricc2` calculations based on foregoing RI-JK Hatree-Fock calculation.

`nozpreopt`
: disable the preoptimization of the Z vector by a preceding RI-CPHF calculation with the cbas basis set. (Note that the preoptimization is automatically deactivated if the `ricc2` calculation is based on a foregoing RI-JK Hatree-Fock calculation.)

Common options for keywords in the data groups `$ricc2`, `$response`, and `$excitations`:

`operators=diplen,dipvel`
: input of operator labels for first-order properties, transition moments, etc. Currently implemented operators/labels are

  `overlap` overlap (charge) operator: the integrals evaluated in the AO basis are $\langle\mu|\nu\rangle$

  `diplen` dipole operator in length gauge: $\langle\mu|r_i^O|\nu\rangle$ with $i = x,\ y,\ z$; the index O indicates dependency on the origin (for expectation values of charged molecules), which in the present version is fixed to $(0, 0, 0)$
  (all three components, individual components can be specified with the labels `xdiplen`, `ydiplen`, `zdiplen`).

  `dipvel` dipole operator in velocity gauge: $\langle\mu|\nabla_i|\nu\rangle$
  (all three components, individual components can be specified with the labels `xdipvel`, `ydipvel`, `zdipvel`).

`qudlen` quadrupole operator $\langle\mu|r_i^O r_j^O|\nu\rangle$
(all six components, individual components can be specified with the labels `xxqudlen`, `xyqudlen`, `xzqudlen`, `yyqudlen`, `yzqudlen`, `zzqudlen`).
If all six components are present, the program will automatically give the electronic second moment tensor (which involves only the electronic contributions) M_{ij}, the isotropic second moment $\alpha = \frac{1}{3}\mathrm{tr}M$ and the anisotropy

$$\beta = \sqrt{\frac{1}{2}\sum_{i=x}^{z}(M_{ii} - M_{i+1,i+1})^2 + 3\sum_{i=x}^{z} M_{i,i+1}^2}\,.$$

Furthermore the traceless quadrupole moment

$$\Theta_{ij} = \frac{1}{2}\langle 3r_i r_j - r^2\delta_{ij}\rangle$$

(including nuclear contributions) is given.

`angmom` angular momentum $\langle\mu|L_i^O|\nu\rangle$
(all three components, individual components can be specified with the labels `xangmom`, `yangmom`, `zangmom`).

`nef` electronic force on nuclei $\langle\mu|\frac{Z_I r_i^I}{r^{I3}}|\nu\rangle$, where Z_I is the charge of the nucleus I and r^I is the position vector of the electron relative to the nucleus (all three components for all nuclei: the labels are `xnef001`, `ynef001`, `znef001`, `xnef002`, etc. where the number depends on the order in the `coord` file).

`states=all`
specification of states for which transition moments or first-order properties are to be calculated. The default is `all`, i.e. the calculations will be done for all excited states for which excitation energies have been calculated. Alternatively, one can select a subset of these listed in parentheses, e.g. `states=(ag{3} 1,3-5; b1u{1} 1-3; b2u4)`. This will select the triplet a_g states no. 1, 3, 4, 5 and the singlet b_{1u} states no. 1, 2, 3 and the singlet (which is default if no `{}` is found) b_{2u} state no. 4.

`$D2-diagnostic`
Calculate the double-substitution-based diagnostics D_2.

`$cc2_natocc`
Write MP2/CC2 natural occupation numbers and natural orbitals to a file.

`$cgrad 1000`
Calculate the error functional δ_{RI} for the RI approximation of $(ai|bj)$ integrals

$$\delta_{\mathrm{RI}} = \frac{1}{4}\frac{|\langle ab||ij\rangle_{exact} - \langle ab||ij\rangle_{\mathrm{RI}}|^2}{\epsilon_a - \epsilon_i + \epsilon_b - \epsilon_j}$$

and its gradients with respect to exponents and coefficients of the auxiliary basis set as specified in the data group `$cbas`. The results are written to `$egrad` scaled by the factor given with the keyword `$cgrad` and can be used to optimize auxiliary basis sets for RI-MP2 and RI-CC2 calculations (see Section 1.5).

14.2.14 Keywords for Module RELAX

`$optimize` *options*

define what kind of nonlinear parameters are to be optimized by `relax` and specify some control variables for parameter update.

Available options are:

`internal on/off`

optimize molecular structures in the space of internal coordinates using definitions of internal coordinates given in `$intdef` as described in Section 4.1 (default: `on`).

`redundant on/off`

optimize molecular structures in redundant internal coordinates using definitions of redundant internal coordinates given in `$redundant`. For an optimization in redundant internal coordinates option `internal` has to be switched `on` too, and option `cartesian` has to be switched `off` (default: `on`).

`cartesian on/off`

optimize molecular structures in the space of (symmetry-distinct) cartesian coordinates (default: `off`).

`basis on/off` *suboptions*

optimize basis set exponents (default=`off`).

Available suboptions are:

`logarithm`

exponents of uncontracted basis functions will be optimized after conversion into their logarithms (this improves the condition of the approximate force constant matrix obtained by variable metric methods and the behavior of the optimization procedure); scale factors of contracted basis functions will not be affected by the logarithm suboption

`scale`

ALL basis set exponents will be optimized as scale factors (i.e. contracted blocks and single functions will be treated in the same way); if both suboptions (scale and logarithm) are given the logarithms of the scale factors will be optimized

`global on/off`

optimize a global scaling factor for all basis set exponents (default: `off`).

NOTES:
- basis and global have to be used exclusively!
- if `$optimize` has been specified but `$forceapprox` is absent, the option `$forceinit on` is switched on by default.
- specification of the option `$interconversion on` will override `$optimize`!

`$coordinateupdate` *options*

define some variables controlling the update of coordinates.

Available options are:

`dqmax` *real*

maximum allowed total change for update of coordinates. The maximum change of individual coordinate will be limited to $dq_{max}/2$ and the collective change dq will be damped by $dq_{max}/\langle \mathrm{d}q | \mathrm{d}q \rangle$ if $\langle \mathrm{d}q | \mathrm{d}q \rangle > dq_{max}$q (default: `0.3`)

`interpolate on/off`

calculate geometry update by inter/extrapolation of geometries of the last two cycles (the interpolate option is always switched on by default, but it is only active ANY time if steepest descent update has been chosen, i.e. `$forceupdate method=none`; otherwise it will only be activated if the DIIS update for the geometry is expected to fail)

`statistics on/`*integer*`/off`

provide a statistics output in each optimization cycle by displaying all (the last *integer*, default setting by `define` is `5`) subsequent coordinates, gradient and energy values (default: `on`).

`$gdiishistory file=`*char*

the presence of this keyword forces `relax` to provide informational output about the usage of DIIS for the update of the molecular geometry.

`$interconversion` *options* `default=off`

special input related to the transformation of atomic coordinates between cartesian and internal coordinate spaces (default: `off`).

Available options are:

`maxiter=`n

maximum number of iterations for the iterative conversion procedure internal → cartesian coordinates (default: 25).

`qconv`

convergence criterion for the coordinate conversion (default: 1.d-10).

`on/off` *options*

this switch activates special tasks: transform coordinates/gradients/hessians between spaces of internal/cartesian coordinates using the definitions of internal coordinates given in `$intdef`:

available suboptions are:

`cartesian --> internal coordinate gradient hessian`

`cartesian <-- internal coordinate`the direction of the transformation is indicated by the direction of the arrow

Note: specification of `$interconversion on` will override `$optimize`!

`$forceupdate` *method options*

this data group defines both the method for updating the approximate force constant matrix and some control variables needed for the force constant update.

Options for **method**:

`none`	no update (steepest descent)
`ms` *suboptions*	Murtagh–Sargent update
`dfp` *suboptions*	Davidon–Fletcher–Powell update
`bfgs` *suboptions*	Broyden–Fletcher–Goldfarb–Shanno update
`dfp-bfgs` *suboptions*	combined (bfgs+dfp) update
`schlegel` *suboptions*	Schlegel update
`ahlrichs` *suboptions*	Ahlrichs update (macro option)

suboptions if `method=ms`, `dfp`, `bfgs`, `schlegel`, `ahlrichs`

`numgeo=`*integer*	number of structures used
`maxgeo=`*integer*	maximum number of geometries (= rank of the update procedure, for `ahlrichs` only)
`ingeo=`*integer*	minimum number of geometries needed to start update

if *method*=`ms`, `dfp`, `bfgs`: `maxgeo=2`, `mingeo=1` as default

additional *suboptions* if `method=ahlrichs`

`modus=` *char fmode*	for an explanation see *suboptions* `pulay` given below e.g. `ahlrichs numgeo=7 mingeo=3 maxgeo=4 modus=<g|dg> dynamic`

NOTES: if the macro option `ahlrichs` has been chosen and n=`numgeo`, *ncycl*='number of geometries available'

- if $ncycl < n$: geometry update by inter/extrapolation using the last two geometries
- if $ncycl \geq n$: diagonal update for the hessian by least mean squares fit; pulay update for the geometry (using specified *modus*, *fmode* (see `pulay` below))
- if ($ncycl \geq \max(5, n+3)$ and $\max(|\,g\,|) < 0.01$ and $\bar{g} < 0.001\,$) or $\mathbf{H}_{ij} \neq 0 \,\forall\, i \neq j$: diagonal update is replaced by multidimensional BFGS (rank n) update for the hessian

`pulay` *suboptions*

try to find an optimal linear combination of the coordinates of the `numpul` previous optimization cycles as specified by `modus` (see below).

Available suboptions are:

`numpul=`*integer*

number of geometries to be utilized

`maxpul=`*integer*

maximum number of geometries

`minpul=`*integer*

minimum number of geometries needed to start update

`modus=`*char fmode*

char=`<g|g>` or `<g|dq>` or `<dq|dq>` defines the quantity to be minimized (`dq` = internal coordinate change).
fmode specifies the force constants to be used (only if *char*=`<g|dq>` or `<dq|dq>`!)
fmode=`static`: use static force constants
fmode=`dynamic`: use updated force constants

`fail=`*real*

real defines the threshold for the quantity $g * \mathrm{d}q/|g| * |\mathrm{d}q|$ which defines the angle between gradient vector and coordinate change (default: `0.1`). If `pulay` is used in connection with a multidimensional BFGS update for the hessian than the default is *real*=`0.0`. If $\frac{g \cdot \mathrm{d}q}{|g| * |\mathrm{d}q|} > -real$ the pulay update for the geometry is expected to fail and will be ignored. For example:

```
pulay numpul=4 maxpul=4 minpul=3 modus=<dq|dq> static fail=0.2
```

options for `$forceupdate`

`diagonal`
: update only the diagonal force constants (update for off-diagonals will be suppressed) (only active if *method*=`ms`, `dfp`, `bfgs`)

`offdamp` *real*
: this allows to damp off-diagonal force constants by 1/*real* (compare offreset, which discards off-diagonals completely). Only values > 1.0 will be accepted. This option is active only within one `relax` run and will be disabled automatically by `relax`. This is useful in difficult cases, where the non-diagonal update has lead to too large non-diagonal elements of the hessian.

`offreset`
: reset off-diagonal force constants to zero. This option will be active for the current optimization cycle only, i.e. it will be removed by `relax` after having discarded off-diagonals!

`allow=`*real*
: optimization cycle specification of a maximum energy change allowed (given in mHartree) which will be accepted using the actual approximate force constant matrix from `$forceapprox`; if this energy change will be exceeded, the force constants will be scaled appropriately
(The default: `0.0` means NO action)

`scale=`*real*
: scaling factor for the input hessian (default: `1.0`).

`threig=`*real*
: lower bound for eigenvalues of the approximate hessian (default: `0.005`); if any eigenvalue drops below `threig`, it will be shifted to a reasonable value defined by:

`reseig=`*real*default: texttt0.005.

`thrbig=`*real*
: upper bound for eigenvalues of the hessian; if any eigenvalue exceeds `thrbig`, it will limited to this value (default: `1000.0`).

`damping=`*real*
: damp the variable metric update for the hessian by 1/(1+ *real*) (default: `0.0`).

`$forceinit` *option*
: specify initialization of the (approximate) force constant matrix.

Available options are:

`on/off`
: this activates or deactivates initialization; if `on` has been set, `relax` will provide an initial force constant matrix as specified by one of the possible initialization options as described below and will store this matrix in data group `$forceapprox`; after initialization `relax` resets `$forceinit` to `off`!

`diag=`*suboptions*

provide a diagonal force constant matrix with:

available suboptions are:

`real`

this will lead to an assignment of diagonal elements (default: `1.0`)).

`default`

this will lead to an assignment of initial force constant diagonals depending on the coordinate type.

`individual`

Provide individual defined force constant diagonals for

- internal coordinates (supplied in `$intdef ... fdiag=..`)
- a global scale factor (`$global ... fdiag=..`)

This does not work for basis set optimization. For the correct syntax of '`fdiag=..`' see descriptions of `$intdef`, `$global`

`carthess`

read a cartesian (e.g. analytical) hessian from `$hessian` and use it as a start force constant matrix; if `$optimize internal` has been set: use its transform in internal coordinate space. If large molecules are to be optimized, it may be necessary (large core memory requirements!) to deactivate the numerical evaluation of the derivative of the B-matrix with respect to cartesian coordinates, which is needed to transform $\mathbf{H}(\mathbf{cart}) \rightarrow \mathbf{H}(\mathrm{int})$ exactly by specifying `no dbdx`.

`$last SCF energy change` = *real*

`$last MP2 energy change` = *real*

These keywords depend on the optimization task to be processed and are updated by the corresponding program (i. g. SCF energy).

`$m-matrix` *options*

This data block contains non-default specifications for the m-matrix diagonals. This is of use if some cartesian atomic coordinates shall be kept fixed during optimization.

Available options are:

integer real real real

atomic index followed by diagonal elements of the m-matrix for this atom

`$scratch files`

The scratch file `ftmp` allocated by `relax` can be placed anywhere in your file systems instead of the working directory by referencing its pathname in this data group as follows:

```
$scratch files
 relax   ftmp          path/file
```

The first column specifies the program, the second column the scratch file and the third column the pathname of the file to be used as scratch file.

Input Data Blocks Needed by RELAX

`$intdef` or `$redundant`
: Definitions of internal coordinates and, optionally, values of internal coordinates (`val=...`, given in a.u. or degrees) or force constant diagonal elements (`fdiag=...`).

`$grad`
: Cartesian coordinates and gradients calculated in subsequent optimization cycles. Entries are accumulated by one of the gradient programs (`grad`, `mpgrad`, `rimp2`, `ricc2`, `egrad`, etc.).

`$egrad`
: Basis set exponents scale factors and their gradients as calculated in subsequent optimization cycles. Entries are accumulated by one of the gradient programs.

`$globgrad`
: Global scale factors and gradients as calculated in subsequent optimization cycles. Entries are accumulated by the `grad` or `aoforce` program.

`$corrgrad`
: Allows to augment internal SCF gradients by approximate increments obtained from treatments (e.g. correlation or relativistic) on higher level. See the example below.

```
$corrgrad
#  coordinate    increment
     1            0.0600
     8           -0.0850
```

`$forceapprox` *options*
: Approximate force constant matrix (as needed for geometry optimization tasks). The storage format may be specified by the
available options:

  `format=`*format*
  : the default format is `format=(8f10.5)`, but other 10-digit f10.x formats (e.g. x=4,6,..) are possible and will be used, after being manually specified within `$forceapprox`. See the example below:

```
$forceapprox  format=(8f10.4)
    0.9124
    -.0108    0.3347
    0.2101    0.0299    1.3347
    0.0076    0.1088    0.0778    0.6515
```

`$hessian (projected)`
: this data block contains the analytical cartesian force constant matrix (with translational and rotational combinations projected out) as output by the `aoforce` program and may be used to supply a high quality force constant matrix `$forceapprox` for geometry optimizations (specifying `$forceinit on carthess`, or `$interconversion cartesian --> internal hessian`).

RELAX Output Data Groups

`$coord`
: either updated cartesian coordinates if a successful coordinate update has been performed, or cartesian coordinates for input internal coordinates if only a conversion from internal to cartesian coordinates has been performed.

`$basis`
: updated basis set exponents, basis sets contraction coefficients or scaling factors, if `$optimize basis on` has been specified.

`$global`
: updated global scaling factor for all basis set exponents, if `$optimize global on` has been specified.

`$forceapprox`
: an approximate force constant matrix to be used in quasi-Newton type geometry optimizations; this matrix will be improved in subsequent optimization cycles if one of the variable-metric methods (`$forceupdate`) has been chosen. See 5.3.13 and 14.2.14.

`$forcestatic`
: a static (i.e. never updated) approximate force constant matrix to be used in DIIS-type geometry optimizations. It will be initialized by `relax` specifying: `$forceupdate pulay ...modus=<dq|dq> static`.

The next data groups are output by `relax` (depending on the optimization subject) in order to control the convergence of optimization procedures driven by the shell script `jobex`.

`$maximum norm of cartesian gradient =` *real*

`$maximum norm of internal gradient =` *real*

`$maximum norm of basis set gradient =` *real*

real is the absolute value of the maximum component of the corresponding gradient.

Other Input/Output data used by RELAX

In order to save the effort for conversion of accumulated geometry and gradient data (as needed for the force constant update or the DIIS update of the geometry) to the optimization space, within which the geometry has to be optimized, one may specify the keyword

`$oldgrad`

Then the `relax` program accumulates all subsequent coordinates and gradient as used in optimization in this data group (or a referenced file). This overrides the input of old coordinate and gradient data from data blocks `$grad`, `$egrad`, ... as accumulated by the `grad` program.

degrees

14.2.15 Keywords for Module STATPT

```
$statpt
  itrvec      0
  update      bfgs
  hssfreq     0
  keeptmode
  hssidiag    0.5
  radmax      0.3
  radmin      1.0d-4
  tradius     0.3
  threchange  1.0d-5
  thrmaxdispl 1.0d-3
  thrmaxgrad  1.0d-3
  thrrmsdispl 5.0d-4
  thrrmsgrad  5.0d-4
```

Only non-default values are written in the `control` file except:

```
$statpt
  itrvec 0
```

Following options are available:

`itrvec`
: Index of the Hessian eigenvector to follow for transition structure search (transition vector). Eigenpairs are sorted in ascending order, i.e. with increasing eigenvalues and start with index 1. The eigenpairs corresponding to translations and rotations are shifted to the end. For minimization the value 0 has to be specified.

`update`
: Method of hessian update. For minimization default is *BFGS*, for TS search default is *Powell* and *none* is for no update.

`hssfreq`
: Frequency of hessian calculation.

`keeptmode`
: Freezing transition vector index.

`hssidiag`
: diagonal hessian elements for diagonal Hessian guess (default: 0.5).

`radmax`
: Maximum allowed value for trust radius (default: 0.3).

`radmin`
: Minimum allowed value for trust radius (default: 1.0d-4).

`tradius`
: Initial value for trust radius (default tradius: radmax = 0.3).

Convergence criteria

`threchange`	threshold for energy change (default: 1.0d-5).
`thrmaxdispl`	threshold for maximal displacement element (default: 1.0d-3).
`thrmaxgrad`	threshold for maximal gradient element (default: 1.0d-3).
`thrrmsdispl`	threshold for RMS of displacement (RMS = root mean square) (default = 5.0d-4)
`thrrmsgrad`	threshold for RMS of gradient (default: 5.0d-4).

All values are in atomic units.

14.2.16 Keywords for Module MOLOCH

`$properties` specifies the global tasks for program `moloch` by virtue of the following options

```
$properties
    trace                          off
    moments                        active
    potential                      off
    cowan-griffin                  off
    localization                   off
    population analyses            off
    plot                           off
    firstorder                     off
    fit                            off
```

a missing option or a option followed by the flag `off` will not be taken into account. The flag `active` may be omitted. For most of these options (with the only exceptions of `trace` and `cowan-griffin`), there are additional data groups allowing for more detailed specifications, as explained below.

`moments`
: if `moment` is active you need

```
$moments
  0th 1st 2nd 3rd
  point .0 .0 .0
```

to compute the 0th, 1st, 2nd and 3rd moment at the reference point 0 0 0.

`potential`
: if `potential` is active you need

```
$points #1
  pot fld fldgrd shld
  point .0 .0 .0
```

to compute the electrostatic potential (`pot`) and/or electrostatic field (`fld`) and/or electrostatic field gradient (`fldgrd`) and/or the zeroth order contribution to the diamagnetic shielding (`shld`) at reference point 0 0 0.

`localization`
: if `localization` is active you need `$boys` to perform a boys-localization of orbitals with orbital energies $\geq$ `thresholad`=-2 Hartrees; localize with respect

to `locxyz`=x, y and z and write resulting orbitals to `lmofile`= 'lmo'. At the most `sweeps`=10000 orbital rotations are performed. Non-defaults may be specified using the following suboptions:

`lmofile=` *filename*

`locxyz` *dir1 dir2 dir3*

`threshold` *real*

`sweeps` *integer*

`population analyses`

if `population analyses` is active you need

```
$mulliken
 spdf molap netto irpspd irpmol mommul
```

to perform a Mulliken population analysis. The options specify the output data:

`spdf`	print molecular orbital contributions to atomic s, p, d,...-populations
`molap`	print molecular orbital contributions to overlap populations
`netto`	print atomic netto populations
`irpspd`	print contributions of (irreducible) representations to atomic s,p,d,...-populations
`irpmol`	print contributions of (irreducible) representations to overlap populations

or

`$loewdin`

to perform a Löwdin population analysis (options are invalid here). A Löwdin population analysis is based on decomposing $\sqrt{\mathbf{S}}\mathbf{D}\sqrt{\mathbf{S}}$ instead of $\mathbf{DS}$ in case of a Mulliken PA.

or

```
$paboon
 momao maodump maofile=mao all
```

to perform a population analysis based on occupation numbers (the options are not necessary and produce some output data concerning the modified atomic orbitals):

`momao`	print MO contributions to occupation numbers of modified atomic orbitals (MAOs).
`maodump`	print all MAOs on standard output

`maofile=mao all`
print all MAOs to file `mao`.

This kind of population analysis basically aims at so-called shared electron numbers (SEN) between two or more atoms. By default 2-, 3- and 4-center contributions to the total density are plotted if they are larger than 0.01 electrons. Thresholds may be individually chosen, as well as the possibility to compute SENs for molecular orbitals: `$shared electron numbers orbitals`

`2-center threshold =` *real*
`3-center threshold =` *real*
`4-center threshold =` *real*

Results of this kind of PA depend on the choice of MAOs. By default, all MAOs with eigenvalues of the atomic density matrices larger than 0.1 will be taken into account. This is a reasonable minimal basis set for most molecules. If modified atomic orbitals shall not be selected according to this criterion, the data group `$mao` selection has to be specified

`$mao selection threshold =`*real*;

The default criterion for the selection of MAOs is the occupation number, for which a global threshold can be specified within the same line as the keyword `$maoselection`. If the global criterion or threshold is not desirable for some atoms, lines of the following syntax have to be added for each atom type of these.

`atom` *symb list* `nmao=`*i* `method=`*meth* `threshold=`*r*

The parameters in this definition have the following meaning:

`symb` atom symbol

`list` list of all atoms for which this definition should apply. The syntax for this list is as usual in `TURBOMOLE`, e.g. `2,3,8-10,12`

`nmao=`*i*
means number of MAOs to be included

`method=`*meth*
means selection criterion for MAOs. *meth* can be `occ` (default), `eig`, or `man` *string*, where `occ` denotes selection of MAOs by occupation numbers, `eig` selection by eigenvalues and `man` allows manual selection. In the latter case the string (max. 8 characters) appended to `man` serves as nickname for the definition of the MAOs to be chosen. This nickname is expected to appear as the leftmost word in a line somewhere within data group `$mao selection` and is followed by the indices of the modified atomic orbitals which are to be selected.

`threshold=`*r*
means the threshold to be applied for the selection criteria `occ` or `eig` (default: 0.1).

Example:

```
$mao selection  threshold= 0.09
   atom c 1,3-5  nmao= 5  method= eig  threshold= 0.1
   atom o 2      nmao= 3  method= man olabel
 olabel  3-5
```

`plot`

option `plot` is out of fashion; to plot quantities on a grid, rather use `$pointval` in connection with `dscf`, `ridft`, `rimp2` or `egrad`, as described below. If nevertheless `plot` is active you need

```
$grid     #1
 mo 4a1g
  origin      .000000      .000000      .000000
 vector1     1.000000      .000000      .000000
 vector2      .000000     1.000000      .000000
 grid1 range    -5.000000     5.000000 points   100
 grid2 range    -5.000000     5.000000 points   100
 outfile = 4a1g
```

to obtain two-dimensional plot data of mo 4a1g (the plane is specified by origin and two vectors with grid range and number of grid points) which is written to file 4a1g. Several plots may be obtained (#1, #2 etc.) at the same time. Use tool 'konto' to visualize the plot.

Note: This is the old-fashioned way to plot MOs and densities. A new—and easier—one is to use `$pointval`, as described below.

`fit`

if `fit` is active you need

```
$vdw_fit
shell     number_of_gridpoints     distance_from_vdW_surface
refine    value_of_potential
```

`shell` Each line refers to all atoms, the line specifies a spherical layer of grid points around the atoms. The number of points and their distance from the van der Waals surface [Bohr] are given (the default is 1.0).

`refine` one line only, smoothing of the layers of grid points around the molecule: the real number is used to define isopotential surfaces on which the points of the layers have to lie.

```
$vdw_radii
element_symbol    van_d_waals_radius
```

One line per element has to be specified, it contains the name of the element and the van der Waals radius in [Bohr].

14.2.17 Keywords for wave function analysis and generation of plotting data

Properties of RHF, UHF and (two-component) GHF wave functions as well as those of SCF+MP2 densities or such from excited state DFT-calculations can be directly analysed within the respective programs (`dscf`, `ridft`, `mpgrad`, `rimp2` and `egrad`). In case of spin-unrestricted calculations results are given for total densities ($D^\alpha + D^\beta$) and spin densities ($D^\alpha - D^\beta$). If not explicitly noted otherwise, in the following "D" is the SCF density, D(SCF), in case of `dscf` and `ridft`, the MP2-corrected density, D(SCF)+D(MP2), for `mpgrad` and `rimp2` and the entire density of the excited state in case of `egrad`. For modules `dscf` and `ridft` the analysis of properties may be directly started by calling `dscf -proper` (or `ridft -proper`). In case of `mpgrad` and `rimp2` this is possible only, if the MP2 density has already been generated, i.e. after a complete run of `mpgrad` or `rimp2`.

Functionalities of analyses are driven by the following keywords.

`$mvd`

leads to calculation of relativistic corrections for the SCF total density in case of `dscf` and `ridft`, for the SCF+MP2 density in case of `rimp2` and `mpgrad` and for that of the calculated excited state in case of `egrad`. Quantities calculated are expectation values $< p^2 >, < p^4 >$ and the Darwin term ($\sum 1/Z_A * \rho(R_A)$).

`$moments`

yields calculation of electrostatic moments arising from nuclear charges and total electron densities. Also without setting this keyword moments up to quadrupole are calculated, with respect to reference point (0,0,0). Supported extensions:

```
$moments <i>
x1 y1 z1
x2 y2 z2
.
.
```

By integer i; the maximum order of moments is specified, maximum and default is i=3 (octopole moments), real numbers x, y, z allow for the specification of one or more reference points.

`$pop`

drives the options for population analyses. By default a Mulliken PA in the basis of cartesian atomic orbitals (CAOs) is performed for the total density ($D^\alpha + D^\beta$) leading to Mulliken (brutto) charges and, in case of spin-unrestricted calculations also for the spin density ($D^\alpha - D^\beta$) leading to Mulliken (brutto) numbers for unpaired electrons. Besides total numbers also contributions from s-, p-, ... functions are listed separately.

Two-component wavefunctions (only module `ridft` and only if `$soghf` is set): In two-component calculations instead of S_z $|(S_x, S_y, S_z)|$ is written to the output. Additionally a vector-file named `spinvec.txt` is written, which includes the resulting spinvector for each atom in the system (also the direction).

The following modifications and extensions are supported, if the respective commands are written in the same line as `$pop`:

`lall` Additional information about p_x, p_y, p_z (and analogous for d and f functions) is displayed (lengthy output).

`atoms` *list of atoms*
Contributions are plotted only if arising from atoms selected by list.

`thrpl`=*real*
Contributions smaller than `thrpl` are not displayed (default: 0.01).

`overlap`Mulliken atomic overlap matrix is displayed.

`netto`Mulliken netto populations (diagonal elements of Mulliken overlap matrix) are calculated.

`mosum` *list of MOs*
Summed Mulliken contributions for a group of molecular orbitals defined by numbers referring to the numbering obtained e.g. from the tool `eiger`. Note that occupancy of MOs is ignored, i.e. all orbitals are treated as occupied.

`mo` *list of MOs*
Mulliken contributions for single MOs defined by numbers (independent of whether they are occupied or not). If this option is valid, one may additionally set

`dos width`=*real* `points`=*integer*
to calculate a (simulated) density of states by broadening the discrete energy levels with Gaussians and superimposing them. The width of each Gaussian may be set by input (default: 0.01 a.u.). The resolution (number of points) may be chosen automatically (default values are usually sufficient to generate a satisfactory plot) or specified by hand. The output files (`dos` in case of RHF wave functions, and `dos_a+b`, `dos_a-b`, `dos_alpha`, `dos_beta`; for UHF cases) contain energies (first column), resulting DOS for the respective energy (second column) as well as s-, p-, d-contributions for the respective energy (following columns).

Example:

```
$pop mo 23-33 dos atoms 2,3,7-8
```

leads to Mulliken PA (CAO-basis) for each of the eleven MOs `23-33`, regarding only contributions from atoms `2-3` and `7-8` (results are written to standard output) and generation of file(s) with the respective simulated density of states.

`$pop nbo`
to perform a natural population analyses [117]. The possible options (specified in the same line) are

`AO`	must be provided, the CAO case is not implemented.
`tw=`*`real`*	Threshold t_w to circumvent numerical difficulties in computing O_w (default: tw=1.d-6).
`idbgl=`*`integer`*	Debug level (default: idbgl=0).
`ab`	For UHF cases: Print alpha and beta density results.
`short`	Print only natural electron configuration and summary.

Example:

```
$pop nbo AO ab short atoms 1,2,6
```

leads to a natural population analysis (AO-basis) with printing the results of alpha and beta densities (only the electron configuration and the summary) for the atoms 1,2 and 6.

To change the NMB set for atoms, one has to add a $nbonmb-block in the *control* file. Example:

```
$nbonmb
  ni s:4 p:2 d:1
  o  s:2 p:1
```

leads to a NMB set for Ni of 4 s-, 2 p- and 1d-functions and for O of 2 s- and 1 p-functions.

`$pop paboon`
to perform a population analyses based on occupation numbers [118] yielding "shared electron numbers (SENs)" and multicenter contributions. For this method always the total density is used, i.e. the sum of alpha and beta densities in case of UHF, the SCF+MP2-density in case of MP2 and the GHF total density for (two-component-)GHF.

The results of such an analysis may depend on the choice of the number of modified atomic orbitals ("MAOs"), which can be specified by an additional

line; without further specification their number is calculated by the method "mix", see below. Note: One should carefully read the information concerning MAOs given in the output before looking at the numbers for atomic charges and shared electron numbers.

`$mao selection` *options*

to specify how MAOs are selected per atom.

Available options are:

a) for the way of sorting MAOs of each atom:

`eig`

MAOs are sorted according to their eigenvalue (those with largest EW finally are chosen). This is the default.

`occ`

MAOs are sorted according to their occupation; note that the number of all occupation is NOT the number of electrons in the system. This option is kept rather for historical reasons.

b) for the determination of the number of MAOs:

`fix`

A fixed number of MAOs is taken for each atom; usually this is the number of shells up to the complete valence shell, e.g. 5 for B-Ne, 6 for Na-Mg, etc. Exceptions are Elements Sc (Y, La), Ti (Zr, Hf), V (Nb, Ta) for which not all five d-shells are included, but only 2, 3 or 4, respectively. This modification leads to beetter agreement with partial charges calculated by an ESP-fit.

`thr <real>`

All MAOs with an eigenvalue larger than `<real>` are chosen; default is `<real>=0.1`. This and the following two options are not valid in connection with `occ`.

`max`

Maximum of numbers calculated from `fix` and `thr=0.1` is taken.

`mix`

2:1 mixture of `fix` and `thr=0.1`. This choice gives best agreement (statistical) with charges from an ESP-fit. It is the default choice.

c) for additional information about MAOs:

`info`

Eigenvalues and occupations for each MAO are written to output.

`dump`

Entire information about each MAO is written to output. Lenghty.

Further for each atom the number of MAOs and the sorting mode can be specified individually in lines below this keyword. Example:

```
atom 1,3-4 eig 7
```

leads to choice of the 7 MAOs with largest eigenvalue at atoms 1, 3-4.

`$localize`

enables the generation of localized molecular orbitals (LMOs) using Boys localization. By default, all occupied orbitals are included, localised orbitals are written (by default in the AO-basis) to file(s) `lmo` in case of RHF and `lalp` and `lbet` in case of UHF orbitals. Note, that LMOs usually break the molecular symmetry; so, even for symmetric cases the AO (not the SAO) basis is used for the output. The localized orbitals are sorted with respect to the corresponding diagonal element of the Fock matrix in the LMO basis. In order to characterize these orbitals, dominant contributions of (canonical) MOs are written to standard output as well as results of a Mulliken PA for each LMO (for plotting of LMOs see option `$pointval`).

The keyword allows for following options (to be written in the same line):

`mo` *list of MOs*
: Include only selected MOs (e.g. valence MOs) in localization procedure (numbering as available from `Eiger`).

`sweeps=`*integer*
: maximum number of orbital rotations to get LMOs; default value is 10000 (sometimes not enough, in particular for highly delocalised systems).

`thrcont=`*real*
: lower threshold for displaying MO and Mulliken contributions (default: 0.1).

`CAO` LMOs are written to file in the CAO basis (instead of AO)

`$esp_fit`

fits point charges at the positions of nuclei to electrostatic potential arising from electric charge distribution (also possible for two-component calculations, for UHF cases also for spin density). For this purpose the ("real") electrostatic potential is calculated at spherical shells of grid points around the atoms. By default, Bragg-Slater radii, r_{BS}, are taken as shell radii, for each atom the number of points is given by $1000 \cdot r_{\mathrm{BS}}^2$, the total number of points is the sum of points for each atom reduced by the number of points of overlapping spheres. Non-default shells (one or more) can be specified as follows:

```
$esp_fit
shell i1 s1
shell i2 s2
⋮
```

Integer numbers i define the number of points for the respective shell, real numbers s constants added to radii (default corresponds to one shell with s=1.0).

A parametrization very close to that by Kollman (U.C. Singh, P.A. Kollman, J. Comput. Chem. 5(2), 129-145 (1984)) may be obtained by

```
$esp_fit kollman
```

Here five shells are placed around each atom with r=1.4*$r_{vdW} + k$, k=0pm, 20pm, 40pm, 60pm, 80pm, and r_{vdW} are the van-der-Waals radii of the atoms.

`$pointval`

drives the calculation of space-dependent molecular quantities at 3D grids, planes, lines or single points. Without further specifications the values of densities are plotted on a three-dimensional grid adapted to the molecular size. Data are deposed to output files (suffix `plt`) that can be visualized directly with the gOpenMol program. In case of RHF-dscf/ridft calculations you get the total density on file `td.plt`, for UHF-dscf/ridft calculations one gets both values for the total density ($D^\alpha + D^\beta$) on `td.plt` and the "spin density" ($D^\alpha - D^\beta$) on `sd.plt`. For mpgrad/rimp2 calculations one gets in the RHF case the total density (D(SCF+MP2)) on `td.plt` and the MP2 contribution on `mp2d.plt` and in the UHF case one obtains the total density ($D^\alpha(SCF + MP2)+D^\beta(SCF+MP2)$) on `td.plt`, the "spin density" ($D^\alpha(SCF+MP2)-D^\beta(SCF+MP2)$) on `td.plt`, and the respective MP2 contributions on files `mp2d.plt` and `mp2sd.plt`. For `egrad` it is similar, just replace in the filenames `mp2` by `e`.

Integration of density (if absolute value greater than `eps`) within a sphere (origin x, y, z, radius r) is performed for

`$pointval integrate` *x y z r eps*

By default the origin is at (0,0,0), the radius is chosen large enough to include the whole 3D box and all contributions are regarded (`eps=0`).

Data different from total and spin densities are generated by following (combinable) settings (to be written in the same line as statement `$pointval`):

`pot` leads to calculation of electrostatic potential arising from electron densities, nuclei and—if present—constant electric fields and point charges. The densities used for calculation of potentials are the same as above; the respective filenames are generated from those of densities by replacement of the "d" (for density) by a "p" (for potential). By "`pot eonly`" only the electronic contribution to the electrostatic potential is calculated.

`fld` calculation of electric field. Note, that for 3D default output format (`.plt`, see below) only norm is displayed. Densities used are the same as above, filenames are generated from those of densities by replacement of "d" (for density) by "f" (for field).

`mo` *list of MO numbers*
calculation of amplitudes of MOs specified by numbers referring to the numbering obtained e.g. from the tool `eiger` in the same format. The respective filenames are self-explanatory and displayed in the output.

Note, that also in MP2 and excited state calculations the HF/DFT ground state orbitals are plotted (and not natural MP2/excited orbitals).

Two-component cases: The density of the spinors specified by numbers referring to the numbering obtained e.g. from the file `EIGS` are visualized. By setting the keyword `minco` also the amplitudes of the spinor-parts are calculated, whose weights (the probability of finding the electron in this part) lie above the threshold.

`lmo` *list of LMO numbers*
calculation of amplitudes of LMOs (previously generated by `$localize`) ordered by the corresponding diagonal element of the Fock matrix in the LMO basis.

`nmo` *list of NMO numbers*
calculation of amplitudes of NMOs (previously generated by
`$natural orbitals file=natural` and
`$natural orbital occupation file=natural`

`dens` has to be set, if additionally to one of the above quantities also the density is to be computed.

Output formats may be specified by e.g. `fmt=xyz` if written to the same line as `$pointval`. Supported are:

`xyz` in case of scalars (density, (L)MO amplitudes, electrostatic potential) this format is: $(x, y, z, f(x, y, z))$. In case of vectors components of the vector and its norm are displayed. This format is valid for all types of grid (3D, plane, line, points, see below), it is the default format in case of calculation of values at single points. Output file suffix is `.xyz`.

`plt` only for 3D, default in this case. Data are written to binary files that can be directly read by gOpenMol. Note, that this output is restricted to scalar quantities; thus in case of vectors (E-field) only the norm is plotted. Output file suffix is `.plt`.

`map` only for 3D. Data are written to ASCII files that can be imported by e.g. gOpenMol. Note, that this output is restricted to scalar quantities; thus in case of vectors (E-field) only the norm is plotted. Output file suffix is `.map`.

`txt` a format compatible with gOpenMol for visualization of vectors v. The format is x, y, z, v_x, v_y, v_z.

`vec` for planes and lines (default in these cases). In case of a line specified by $\alpha \cdot \vec{v}$ (see below) output is $\alpha, f(x, y, z)$ for scalars, for vectors components and norm are displayed. vectors. Analogously, in case of planes it is $\alpha, \beta, f(x, y, z)$. The output (file suffix `.vec`) may be visualized by plotting programs suited for two-dimensional plots. A command file (termed `gnuset`) to get a contour plot by gnuplot is automatically generated.

`cub` only for 3D, writes out data in Cube format which can be imported by many external visualization programs.

For 3D grids non-default boundarys, basis vector directions, origin and resolutions may be specified as follows:

```
$pointval
grid1 vector 0 3 0 range -2,2 points 200
grid2 vector 0 0 -7 range -1,4 points 300
grid3 vector 1 0 0 range -1,1 points 300
origin 1 1 1
```

Grid vectors (automatically normalised) now are $(0,1,0)$,$(0,0,-1)$,$(1,0,0)$, the grid is centered at $(1,1,1)$, and e.g. for the first direction 200 points are distributed between -2 and 2.

Grids of lower dimensionality may be specified (in the same line as `$pointval`) by typing either `geo=plane` or `geo=line` or `geo=point` The way to use is best explained by some examples:

```
$pointval geo=plane
grid1 vector 0 1 0 range -2,2 points 200
grid2 vector 0 0 1 range -1,4 points 300
origin 1 1 1
```

Values are calculated at a plane spanned by vectors (0,1,0) and (0,0,1) centered at (1,1,1).

```
$pointval geo=line
grid1 vector 0 1 0 range -2,2 points 50
origin 0 0 1
```

Values are calculated at a line in direction (0,1,0) centered at (0,0,1). Output format as above.

```
$pointval geo=point
7 5 3
0 0 7
```

Values are calculated at the two points $(7.0, 5.0, 3.0)$ and $(0.0, 0.0, 7.0)$.

14.2.18 Keywords for Module FROG

The *ab initio* molecular dynamics (MD) program `frog` needs a command file named `mdmaster`. The interactive `Mdprep` program manages the generation of `mdmaster`

and associated files. It is always a good idea to let `Mdprep` check over `mdmaster` before starting an MD run. `Mdprep` has online-help for all menus.

In this implementation of *ab initio* MD, time is divided into steps of equal duration Δt. Every step, the energy and its gradient are calculated and these are used by the `frog` to work out the new coordinates for the next step along the dynamical trajectory. Both the accuracy of the trajectory and the total computation time thus depend crucially on the time step chosen in `Mdprep`. A bad choice of timestep will result in integration errors and cause fluctuations and drift in the total energy. As a general rule of thumb, a timestep Δt should be chosen which is no longer than one tenth of the shortest vibrational period of the system to be simulated.

Note that `Mdprep` will transform velocities so that the total linear and angular momentum is zero. (Actually, for the Leapfrog algorithm, initial velocities are $\Delta t/2$ before the starting time).

The following keywords are vital for `frog`:

`$nsteps` 75
: Number of MD time steps to be carried out. `$nsteps` is decreased by 1 every time `frog` is run and JOBEX -md stops when `$nsteps` reaches 0.

`$natoms` 9
: Number of atoms in system.

`$current file=mdlog.aa`
: The file containing the current position, velocity, time and timestep, that is, the **input** configuration. During an MD run the `$current` information is generally kept at the end of the `$log` file.

`$log file=mdlog.ZZ`
: The file to which the trajectory should be logged, i.e. the **output**: t=time (a.u.);
atomic positions x,y,z (Bohr) and symbols at t;
timestep (au) Δt;
atomic symbols and velocities x,y,z (au) at $t - (\Delta t/2)$;
kinetic energy (H) interpolated at t, *ab initio* potential energy (H) calculated at t, and pressure recorded at the barrier surface (atomic units, 1 au = 29.421 TPa) during the corresponding timestep;
ab initio potential energy gradients x,y,z (H/Bohr) at t.
This file can be manipulated with LOG2? tools after the MD run (Section 1.5).

`$turbomole file=control`
: Where to look for `TURBOMOLE` keywords `$grad` etc.

`$md_status`
: The status of the MD run is a record of the action carried out during the previous MD step, along with the duration of that step. The format matches that of `$md_action` below.

Canonical dynamics is supported using the Nosé-Hoover thermostat. This option can be enabled in `Mdprep` or by the following syntax:

```
$md_status
  canonical T=500 t=100
  from t= -25.0000000000       until t=  0.00000000000
```

Here, T specifies the temperature of the thermostat in K (500 K in the example) and t specifies the thermostat relaxation time in a.u. (100 a.u. in the example). It is advisable to choose the thermostat relaxation 2-10 times larger than the time step. Note that user-defined actions are presently not supported in canonical dynamics mode.

These are optional keywords:

`$seed -123`
Integer random number seed

`$title`
Arbitrary title

```
$log_history
 100                mdlog.P
 71                 mdlog.Q
```

```
$ke_control
  length        50
  response      1
```

To determine the trends in kinetic energy and total energy (average values and overall drifts) it is necessary to read the history of energy statistics over the recent MD steps. The number of MD steps recorded so far in each log file are therefore kept in the `$log_history` entry: this is updated by the program each step. The length of records needed for reliable statistics and the number of steps over which changes are made to kinetic energy (`response`) are specified in `$ke_control`.

```
$barrier angstroms
  type        elps
  limits      5.0 10.0 7.5
  constant    2.0
  thickness   1.0
  temperature 300.0
```

`$barrier` specifies a virtual cavity for simulating condensed phases. The optional flag, `angstroms`, can be used to indicate that data will be entered in Ångstrøms rather than Bohr.

`type`
: can be one of `orth`, `elps`, or `none`, for orthorhombic, ellipsoidal, or no barrier (the default) respectively.

`limits`
: are the +x,y,z sizes of the cavity. In this case, an ellipsoid with a major axis of 20 Å along y, semi-major of 15 Å on z, and minor of 10 Å on x.

`constant`
: is the Hooke's Law force constant in atomic units of force (H/Bohr) per length unit. Here, it is 2.0 H/Bohr/Ångstrøm, a bastard combination of units.

`thickness`
: is the effective limit to the restorative force of the barrier. For this system, an atom at 5 Å into the barrier will feel the same force as at 1.0 Å.

`temperature`
: denotes the temperature of the cavity walls in Kelvin. If the system quasi-temperature is below this setpoint, particles will be accelerated on their return to the interior. Alternately, they will be retarded if the system is too warm. A `temperature` of 0.0 K will turn off wall temperature control, returning molecules to the system with the same momentum as when they encountered the barrier.

```
$constraints angstroms
  tolerance   0.05
  adjpercyc   0.25
  type H O 0.9 1.2
  type F C 0.0 1.7
  type H C -1.0 1.2
  2 1 0.0
  3 1 1.54
  4 1 -1.0
```

`$constraints`
: specifies and/or automatically generates atomic distance constraints. The optional flag, `angstroms`, can be used to indicate that data will be entered in Ångstrøms rather than Bohr.

`tolerance`
: is the convergence criterion for application of constraints. All distances

must be within +/- `tolerance` of the specified constraint. Additionally, the RMS deviation of all constrained distances must be below 2/3 of `tolerance`.

`adjpercyc`
: is the fraction of the total distance correction to be applied on each constraint iteration.

`type X A normalfont` *`const rmax`*
: commands `frog` to find the closest `A` atom to each atom `X` that is closer than `rmax` and apply `const`. The first `type` line above examines each `H` atom and looks for any `O` atoms within 1.2 Å. The shortest distance, if any, is then fixed at 0.9 Å. Similarly, the second `type` line binds each `F` to the closest `C` within 1.7 Å, but since `const`=0.0, that distance is fixed at the current value. The third `type` line attaches `H` atoms to the appropriate nearby `C`, but at the current average `H-C` distance multiplied by the absolute value of `const`.

Explicitly specified constraints are listed by atom index and supercede auto-generated constraints. A positive third number fixes the constraint at that value, while zero fixes the constraint at the current distance, and a negative number unsets the constraint.

The output of `frog` contains the full list of constrained atom pairs and their current constraints in explicit format.

User-defined instructions allow the user to tell `frog` to change some aspect of the MD run at some point in time `t=`*real number*. The same format is used for `$md_status` above. Here is an example:

```
$md_action
     fix total energy from t=2000.0
     anneal from t=2500.0
     free from t=3000.0
```

In this example, starting from the time 2000.0 a.u., velocities are to be scaled every step to keep average total energy constant. Then, from 2500.0 a.u., gradual cooling at the default rate (annealing) is to occur until the time 3000.0 a.u., when free Newtonian dynamics will resume.

Here are all the possible instructions:

```
$md_action
     fix temperature from t=<real>
     fix total energy from t=<real>
```
These commands cause velocities to be scaled so as to keep the average kinetic energy (i.e. quasi-temperature) or the average total energy approximately constant. This is only possible once enough information about run history is available to give reliable statistics. (Keywords `$log_history`, `$ke_control`).

`$md_action`

`set temperature at t=<real> to x=<real> K`

`set total energy at t=<real> to x=<real> H`

`set kinetic energy at t=<real> to x=<real> H`

`set position file=<filename> at t=<real>`

`set velocity file=<filename> at t=<real>`

`set velocity at t=<real> random`

`set velocity at t=<real> zero`

At some time during the *ab initio* MD run the user can specify a new value for one of the dynamical variables. The old value is discarded. Single values are given by `x=`*real number*. Vectors must be read in `frog` format from `file=`*file*.

`$md_action`

`anneal from t=<real>`

`anneal from t=<real> x=<real>`

`quench from t=<real>`

`quench from t=<real> x=<real> file=<file>`

`relax at t=<real>`

In Simulated Annealing MD, the temperature of a run is lowered so as to find minimum-energy structures. Temperature may be lowered gradually by a small factor each step (`anneal`; default factor 0.905 over 100 steps) or lowered rapidly by reversing all uphill motion (`quench`; default factor -0.8 each step). The cooling factors may be changed from the default using `x=`. Another option allows the quenching part of the run to be logged to a separate file. Alternatively, a standard non-dynamical geometry optimisation can be carried out in a subdirectory (`relax`).

`$md_action`

`free from t=<real>`

Finally, this instruction turns off any previous action and resumes free dynamics. This is the default status of an MD run.

14.2.19 Keywords for Module MPSHIFT

In order to control the program execution, you can use the following keywords within the `control` file:

`$csmp2`

Switches on the calculation of the MP2 NMR shieldings. The required SCF shielding step will be performed in the same run. This flag will be set by the script `mp2prep`.

`$traloop` *n*
: specifies the number of loops (or 'passes') over occupied orbitals *n* when doing an MP2 calculation: the more passes the smaller file space requirements—but CPU time will go up. This flag will be set by the script `mp2prep`.

`$mointunit`
: Scratch file settings for an MP2 calculation. Please refer to Section 14.2.12 for a description of the syntax. This flag will be set by the script `mp2prep`.

`$csconv` *real*
: Sets the convergence threshold for the shielding constant of succeeding CPHF iterations. The unit is ppm and the default value is 0.01.

`$csconvatom` *integer*
: This selects the atom number for convergence check after each cphf iteration. After this convergence is reached all other atoms are checked, too (default: 1).

`$thime`, `$thize`, `$scftol`, `$scfintunit`, `$scfmo`
: have the save meaning as in `dscf` (see Section 14.2.5);
Since mpshift works 'semi-direct' it uses the same integral storage.

`$scratch files`
: The scratch files allocated by mpshift can be placed anywhere in your file systems instead of the working directory by referencing their pathnames in this data group. All possible scratch files are listed in the following example:

```
$scratch files
    mpshift   csssmat         path1/file1
    mpshift   cshsmat         path2/file2
    mpshift   csdgsmat        path3/file3
    mpshift   csusmat         path4/file4
    mpshift   dens            path5/file5
    mpshift   fock            path6/file6
    mpshift   dfock           path7/file7
    mpshift   idvds1          path8/file8
    mpshift   idvds2          path9/file9
    mpshift   idvds3          path10/file10
    mpshift   jdvds1          path11/file11
    mpshift   jdvds2          path12/file12
    mpshift   jdvds3          path13/file13
    mpshift   cshmmat         path14/file14
```

`$trast` , `$trand` *traloop-number*
: stands for traloop start and traloop end. Each loop or pass in MP2 chemical shift calculations can be done individually by providing the keywords `$trast`

and `$trand`. This can be used to do a simple parallelization of the run: Create separate inputs for each traloop. Add

```
$trast <number>
$trand <number>
```

in the `control` files, *number* goes from 1 to the number of `$traloops`. Each calculation will create a restart file called `restart.mpshift`. To collect all steps and to do the remaining work, copy all restart files to one directory and rename them to `restart.mpshift.`*number*, add `$trast -1` and `$trand` *number_of_traloops* to the `control` file and start `mpshift`.

14.2.20 Keywords for Parallel Runs

On all systems the parallel input preparation is done automatically. Details for the parallel installation are given in Section 3.2.1. The following keywords are necessary for all parallel runs:

`$parallel_platform` *architecture*
`$numprocs` *number CPUs*

Currently the following parallel platforms are supported:

`SMP` for systems with very fast communication; all CPUs are used for the linear algebra part. Synonyms for `SMP` are:
`HP V-Class`, `SP3-SMP` and `HP S/X-Class`

`MPP` for systems with fast communication like Fast-Ethernet, the number of CPUs that will be taken for linear algebra part depends on the size of the matrices. Synonyms for `MPP` are:
`SP3` and `linuxcluster`

`cluster` for systems with slow communication, the linear algebra part will be done on one single node. Synonyms for `cluster` are:
`HP Cluster` and every platform that is not known by `TURBOMOLE`

`SGI` similar to `SMP`, but here the server task is treated differently: the MPI implementation on the SGIs would cause this task to request too much CPU time otherwise.

`$numprocs` is the number of slaves, i.e. the number of nodes doing the parallel work. If you want to run `mpgrad`, `$traloop` has to be equal to or a multiple of `$numprocs`.

For very large parallel runs it may be impossible to allocate the scratch files in the working directory. In this case the `$scratch files` option can be specified; an example for a `dscf` run is given below. The scratch directory must be accessible from all nodes.

```
$scratch files
   dscf  dens       /home/dfs/cd00/cd03_dens
   dscf  fock       /home/dfs/cd00/cd03_fock
   dscf  dfock      /home/dfs/cd00/cd03_dfock
   dscf  ddens      /home/dfs/cd00/cd03_ddens
   dscf  xsv        /home/dfs/cd00/cd03_xsv
   dscf  pulay      /home/dfs/cd00/cd03_pulay
   dscf  statistics /home/dfs/cd00/cd03_statistics
   dscf  errvec     /home/dfs/cd00/cd03_errvec
   dscf  oldfock    /home/dfs/cd00/cd03_oldfock
   dscf  oneint     /home/dfs/cd00/cd03_oneint
```

For all programs employing density functional theory (DFT) (i.e. `dscf`/`grad`and `ridft`/`rdgrad`) `$pardft` can be specified:

```
$pardft
       tasksize=1000
       memdiv=0
```

The `tasksize` is the approximate number of points in one DFT task (default: 1000) and `memdiv` says whether the nodes are dedicated exclusively to your job (memdiv=1) or not (default: memdiv=0).

For `dscf` and `grad` runs you need a parallel statistics file which has to be generated in advance. The filename is specified with
`$2e-ints_shell_statistics    file=DSCF-par-stat`
or
`$2e-ints'_shell_statistics    file=GRAD-par-stat`
respectively.

The statistics files have to be generated with a single node `dscf` or `grad` run. For a `dscf` statistics run one uses the keywords:

```
$statistics  dscf parallel
$2e-ints_shell_statistics    file=DSCF-par-stat
$parallel_parameters
       maxtask=400
       maxdisk=0
       dynamic_fraction=0.300000
```

and for a `grad` statistics run:

```
$statistics  grad parallel
$2e-ints'_shell_statistics    file=GRAD-par-stat
```

```
$parallel_parameters
       maxtask=400
```

`maxtask` is the maximum number of two-electron integral tasks,
`maxdisk` defines the maximum task size with respect to mass storage (MBytes) and `dynamic_fraction` is the fraction of two-electron integral tasks which will be allocated dynamically.

For parallel `grad` and `rdgrad` runs one can also specify:

```
$grad_send_dens
```

This means that the density matrix is computed by one node and distributed to the other nodes rather than computed by every slave.

In the parallel version of `ridft`, the first client reads in the keyword `$ricore` from the `control` file and uses the given memory for the additional RI matrices and for RI-integral storage. All other clients use the same amount of memory as the first client does, although they do not need to store any of those matrices. This leads to a better usage of the available memory per node. But in the case of a big number of auxiliary basis functions, the RI matrices may become bigger than the specified `$ricore` and all clients will use as much memory as those matrices would allocate even if that amount is much larger than the given memory. To omit this behaviour one can use:

`$ricore_slave` *integer*

specifying the number of MBs that shall be used on each client.

For parallel `jobex` runs one has to specify all the parallel keywords needed for the different parts of the geometry optimization, i.e. those for `dscf` and `grad`, or those for `ridft` and `rdgrad`, or those for `dscf` and `mpgrad`.

Chapter 15

Sample `control` files

15.1 Introduction

The file `control` is the input file for `TURBOMOLE` which directly or by cross references provides the information necessary for all kinds of runs and tasks. `control` is usually generated by `define`, the input generator. The following sample `control` files cover a variety of methods and systems. The keywords themselves are explained in Chapter 14.

15.2 NH_3 Input for a RHF Calculation

Main File control

```
$title
NH3 c3v SVP
$operating system unix
$symmetry c3v
$coord    file=coord
$intdef    file=coord
$atoms
n  1                                                    \
   basis =n def-SVP
h  2-4                                                  \
   basis =h def-SVP
$pople   AO
$basis    file=basis
$rundimensions
   dim(fock,dens)=495
   natoms=4
   nshell=15
   nbf(CAO)=30
   nbf(AO)=29
   dim(trafo[SAO<-->AO/CAO])=69
   rhfshells=1
$scfmo    file=mos
$closed shells
 a1      1-3                                    ( 2 )
 e       1                                      ( 2 )
$scfiterlimit       30
$scfconv        7
$thize      .10000000E-04
$thime        5
$scfdamp   start= .500  step= .050  min= .100
$scfdump
$scfintunit
 unit=30       size=0        file=twoint
$scfdiis   start=0.5
$drvopt
   cartesian  on
   basis      off
   global     off
   hessian    on
```

```
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-10
   maxiter=25
$optimize
   internal   on
   cartesian  off
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.1
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=grad
$forceapprox    file=force
$lock off
$last step     define
$end
```

File coord

```
$coord
     .00000000000000       .00000000000000       .54561506935122      n
    -.87806233111566      1.52084856970468      -.18187168978374      h
    -.87806233111566     -1.52084856970468      -.18187168978374      h
    1.75612466223131       .00000000000000      -.18187168978374      h
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    4    1          val=   1.90084
   2 k  1.0000000000000 bend    4    3    1     val= 106.27756
        1.0000000000000 bend    3    2    1
        1.0000000000000 bend    2    4    1
$end
```

File basis

```
$basis
*
n def-SVP
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
*
   5  s
  1712.8415853     -.53934125305E-02
  257.64812677     -.40221581118E-01
  58.458245853     -.17931144990
  16.198367905     -.46376317823
  5.0052600809     -.44171422662
   1  s
  .58731856571      1.0000000000
   1  s
  .18764592253      1.0000000000
   3  p
  13.571470233     -.40072398852E-01
  2.9257372874     -.21807045028
  .79927750754     -.51294466049
   1  p
  .21954348034      1.0000000000
   1  d
  1.0000000000      1.0000000000
*
h def-SVP
# h     (7s) / [3s]     {511}
*
   3  s
  13.010701000      .19682158000E-01
  1.9622572000      .13796524000
  .44453796000      .47831935000
   1  s
  .12194962000      1.0000000000
   1  p
  .80000000000      1.0000000000
*
$end
```

File mos

```
$scfmo    expanded   format(4d20.14)
```

```
     1  a1      eigenvalue=-.15633041862301D+02   nsaos=10
 .98699003163455D+00-.47221435341751D-01 .55873125006179D-02-.48016374887169D-02
 .26746008768233D-02 .20823779196149D-03 .14270460008808D-01 .90849517503597D-02
 .58676121352806D-03 .29091871198884D-03
     2  a1      eigenvalue=-.99896275238736D+00   nsaos=10
 .26412162337482D+00 .51846472345768D+00 .37623729061179D+00-.77139882704089D-02
-.47252329287316D-02-.21494050853221D-02 .11795673774658D+00 .83316086019184D-01
-.11229203933488D-01-.27038186251429D-02
     3  a1      eigenvalue=-.57101279949392D+00   nsaos=10
-.35584199011701D-01-.96938258881594D-01-.70254605702716D-01 .65569041318341D+00
-.44746149963029D+00 .40094287741992D-03 .51691151834284D-01 .47722350097160D-01
 .19189122068531D-02 .56638497851180D-03
     1  e       eigenvalue=-.64374209294851D+00   nsaos=9
-.49313475446075D+00 .33757893447603D+00-.76142296567409D-04-.74524664248740D-04
-.26407572210452D+00-.22619038902975D+00-.50035170531670D-05-.12199166245418D-03
 .63021657663245D-04
$end
```

15.3 NO_2 input for an unrestricted DFT calculation

Main File control

```
$title
NO2 c2v UKS SVP
$operating system unix
$symmetry c2v
$coord    file=coord
$intdef    file=coord
$atoms
n  1                                                    \
   basis =n def-SVP
o  2-3                                                  \
   basis =o def-SVP
$pople   AO
$basis    file=basis
$rundimensions
   dim(fock,dens)=1098
   natoms=3
   nshell=18
   nbf(CAO)=45
   nbf(AO)=42
   dim(trafo[SAO<-->AO/CAO])=85
   rhfshells=2
$uhfmo_alpha none   file=alpha
$uhfmo_beta none  file=beta
# none : hamilton core guess will be made
# files alpha and beta will be generated by the program
$uhf
$alpha shells
 a1      1-6                                    ( 1 )
 a2      1                                      ( 1 )
 b1      1-4                                    ( 1 )
 b2      1                                      ( 1 )
$beta shells
 a1      1-5                                    ( 1 )
 a2      1                                      ( 1 )
 b1      1-4                                    ( 1 )
 b2      1                                      ( 1 )
$scfiterlimit       30
$scfconv        7
$thize      .10000000E-04
```

```
$thime        5
$scfdamp   start=1.500  step= .050  min= .100
$scfdump
$scfintunit
 unit=30       size=2        file=/work/user/twoint
$scfdiis   start=0.5
$scforbitalshift  closedshell=.3
$drvopt
   cartesian  on
   basis      off
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-10
   maxiter=25
$optimize
   internal   on
   cartesian  off
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.1
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=grad
$forceapprox    file=force
$lock off
$dft
 functional b-p
 gridsize m3
$last step     define
$end
```

File coord

```
$coord
     .00000000000000       .00000000000000     -1.00494155217173      n
    1.85766051386774       .00000000000000       .50247077608587      o
   -1.85766051386774       .00000000000000       .50247077608587      o
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    2    1           val=   2.39232
   2 d  1.0000000000000 stre    3    1           val=   2.39232
   3 k  1.0000000000000 bend    2    3    1      val= 101.88429
$end
```

File basis

```
$basis
*
n def-SVP
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
*
   5  s
  1712.8415853     -.53934125305E-02
  257.64812677     -.40221581118E-01
  58.458245853     -.17931144990
  16.198367905     -.46376317823
  5.0052600809     -.44171422662
   1  s
  .58731856571      1.0000000000
   1  s
  .18764592253      1.0000000000
   3  p
  13.571470233     -.40072398852E-01
  2.9257372874     -.21807045028
  .79927750754     -.51294466049
   1  p
  .21954348034      1.0000000000
   1  d
  1.0000000000      1.0000000000
*
o def-SVP
# o     (7s4p1d) / [3s2p1d]     {511/31/1}
*
   5  s
```

```
  2266.1767785      -.53431809926E-02
  340.87010191      -.39890039230E-01
  77.363135167      -.17853911985
  21.479644940      -.46427684959
  6.6589433124      -.44309745172
   1   s
  .80975975668       1.0000000000
   1   s
  .25530772234       1.0000000000
   3   p
  17.721504317       .43394573193E-01
  3.8635505440       .23094120765
  1.0480920883       .51375311064
   1   p
  .27641544411       1.0000000000
   1   d
  1.2000000000       1.0000000000
*
$end
```

15.4 $TaCl_5$ Input for an RI-DFT Calculation with ECPs

Main File control

```
$title
$operating system unix
$symmetry d3h
$coord    file=coord
$intdef    file=coord
$atoms
ta 1                                                          \
   jbas=ta def-SVP                                            \
   basis =ta def-SVP                                          \
   ecp   =ta def-ecp
cl 2-6                                                        \
   jbas=cl def-SVP                                            \
   basis =cl def-SVP
$pople   AO
$basis    file=basis
$ecp    file=basis
$rundimensions
   dim(fock,dens)=7662
   natoms=6
   nshell=51
   nbf(CAO)=122
   nbf(AO)=115
   dim(trafo[SAO<-->AO/CAO])=346
$scfmo   none   file=mos
# none : hamilton core guess will be made
# file mos will be generated by the program
$scfiterlimit       30
$scfconv        6
$thize      .10000000E-04
$thime        5
$scfdamp   start= .900  step= .050  min= .100
$scfdump
$scfintunit
 unit=30       size=0        file=twoint
$scfdiis   start=0.5
$drvopt
   cartesian  on
   basis      off
   global     off
```

```
   hessian    on
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-10
   maxiter=25
$optimize
   internal   on
   cartesian  off
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.1
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=grad
$forceapprox    file=force
$lock off
$dft
   functional b-p
   gridsize   m3
$last step     define
$ricore    20
$ridft
$jbas file=auxbasis
$closed shells
 a1'     1-11                                   ( 2 )
 a2'     1-2                                    ( 2 )
 e'      1-10                                   ( 2 )
 a2"     1-8                                    ( 2 )
 e"      1-4                                    ( 2 )
$end
```

File coord

```
$coord
     .00000000000000       .00000000000000       .00000000000000      ta
```

```
    2.19392179448315     -3.79998401587749       .00000000000000      cl
    2.19392179448315      3.79998401587749       .00000000000000      cl
   -4.38784358896629       .00000000000000       .00000000000000      cl
     .00000000000000       .00000000000000      4.46615918865523      cl
     .00000000000000       .00000000000000     -4.46615918865523      cl
$intdef
# definitions of internal coordinates
   1 k  1.0000000000000 stre    1    2           val=   4.38784
   2 k  1.0000000000000 stre    1    5           val=   4.46616
$end
```

File basis

```
$basis
*
ta def-SVP
# ta    (7s6p5d) / [6s3p2d]     {211111/411/41}
*
   2  s
  14.400000000      .99343296745
  12.000000000     -1.6510077975
   1  s
  5.0701477302      1.0000000000
   1  s
  .86033356487      1.0000000000
   1  s
  .37158938894      1.0000000000
   1  s
  .10745336254      1.0000000000
   1  s
  .39142776556E-01  1.0000000000
   4  p
  7.4188720000      .26979695152
  5.6984100000     -.46968874449
  1.1777211960      .50905100155
  .54478533555      .52298161137
   1  p
  .22309270117      1.0000000000
   1  p
  .43100000000E-01  1.0000000000
   4  d
  3.9738796278     -.52799310714E-01
  1.4528884813      .18558319471
```

```
  .61042908544      .42959071631
  .24216276510      .43497228232
   1  d
  .87909318337E-01  1.0000000000
*
cl def-SVP
# cl    (7s5p) / [6s2p]     {211111/41}
*
   5  s
  10449.827566      .19708362484E-02
  1571.7365221      .14754727977E-01
  357.12065523      .66679112875E-01
  100.25185935      .17228924084
  30.812727554      .15883786100
   3  s
  51.923789434     -.10009298909
  5.7045760975      .60841752753
  2.3508376809      .54352153355
   1  s
  .44605124672      1.0000000000
   1  s
  .16848856190      1.0000000000
   5  p
  307.66790569     -.87801484118E-02
  72.102015515     -.63563355471E-01
  22.532680262     -.24016428276
  7.8991765444     -.47798866557
  2.8767268321     -.38515850005
   1  p
  .77459363955      1.0000000000
   1  p
  .21037699698      1.0000000000
   1  d
  .65000000000      1.0000000000
*
$ecp
*
ta def-ecp
*
  ncore =    60           lmax =     3
#        coefficient   r^n          exponent
f
          12.0179609    2           2.0178811
```

```
s-f
        1345.8806470    2          14.5464077
          36.7668062    2           7.2732038
         -12.0179609    2           2.0178811
p-f
         378.4253015    2           9.9355653
          22.2930909    2           4.9677824
         -12.0179609    2           2.0178811
d-f
         104.8839557    2           6.3473769
           8.7558481    2           3.1736885
         -12.0179609    2           2.0178811
*
$end
```

File auxbasis

```
$jbas
*
ta def-SVP
*
    3  s
     15.521335           -.493702989D+00
     7.555743             .259256574D+01
     3.699576            -.523168657D+01
    1  s
     1.820141             .262393615D+01
    1  s
     0.898838             .157711902D+01
    1  s
     0.445062             .200789711D+00
    1  s
     0.220729             .185974307D+00
    1  s
     0.109530             .765184411D-01
    1  p
     1.5024958           1.0
    1  p
     0.5629855           1.0
    1  p
     0.2281880           1.0
    1  p
     0.09507835          1.0
```

```
    2  d
     1.337006             .190072032D-01
     0.599535            -.155214344D-01
    1  d
     0.280427            -.138946250D-01
    1  d
     0.133078            -.895263676D-02
    1  f
     1.1428211           1.0
    1  f
     0.4395465           1.0
    1  f
     0.1758186           1.0
    3  g
     1.630421             .100251139D+00
     0.747093             .737448223D-01
     0.349040             .276219913D-01
    1  g
     0.164143             .546316580D-02
*
*
cl def-SVP
*
    8  s
     4097.080409            .198054511D+01
     1203.083193            .530973450D+01
     386.280948             .132352655D+02
     135.337690             .107149960D+02
     51.567046             -.132565114D+01
     21.261034              .271180364D+01
     9.420135               .754640511D+01
     4.445228               .173603618D+01
    1  s
     2.209399              -.140197496D+01
    1  s
     1.141575               .982719736D+00
    1  s
     0.604182               .464178589D+00
    1  s
     0.322378               .369336889D+00
    4  p
     51.8499902611          .359335506D-01
     17.5847835188         -.869599318D-01
```

```
     6.49227239618          .721211200D-01
     2.55889114714         -.634201864D-01
    1  p
     1.05118767781          .264152293D-01
    1  p
      .437994865757         -.197670692D-01
    4  d
     34.705550             -.548703710D-01
     10.704427             -.619019402D-02
     3.568067               .337450480D-01
     1.249848              -.905232209D-01
    1  d
     0.445360               .418680075D-01
    1  f
     1.1872146118          1.0000000
    1  g
     1.30000000            1.0000000
*
$end
```

15.5 Basisset optimization for Nitrogen

Main File `control`

```
$title
 Basisset-optimization for nitrogen SV(P)
$operating system unix
$symmetry oh
#--- uncomment following line to clean the basis-file after optimization ----
#$dump basis set
$coord    file=coord
$user-defined bonds    file=coord
$pople   AO
$basis    file=basis
$rundimensions
   dim(fock,dens)=141
   natoms=1
   nshell=6
   nbf(CAO)=15
   nbf(AO)=14
   dim(trafo[SAO<-->AO/CAO])=17
   rhfshells=2
$scfmo none    file=mos
$roothaan         1
      a = 1      b = 2
$scfiterlimit       60
$scfconv       10
$thize     0.10000000E-04
$thime        5
$scfdamp   start=1.500  step=0.050  min=0.100
$scfdump
$scfintunit
 unit=30       size=90       file=twoint
$scfdiis   start=0.5
$scforbitalshift  closedshell=.4
$drvopt
   cartesian  off
#---- optimize basis! -> basis on ----
   basis      on
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
```

```
$interconversion  off
   qconv=1.d-7
   maxiter=25
$optimize
   internal   off
   cartesian  off
   global     off
#---- optimize basis! -> basis on logarithm ----
   basis      on logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.6
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=gradient
#---- optimize basis! -> $egrad file=egradient ----
$egrad    file=egradient
$forceapprox    file=forceapprox
$lock off
$atoms
n  1                                                                          \
   basis =n def-SV(P)
$closed shells
 a1g     1-2                                    ( 2 )
$open shells type=1
 t1u     1                                      ( 1 )
$end
```

File coord

```
$coord
    0.00000000000000      0.00000000000000      0.00000000000000      n
$user-defined bonds
$end
```

File basis

```
sis
```

```
*
n def-SV(P)
# n     (7s4p1d) / [3s2p1d]     {511/31/1}
# use expopt to optimize exponents and contopt to optimize contractions
*
   5  s    expopt  contopt
  1712.8415853      0.53934125305E-02
  257.64812677      0.40221581118E-01
  58.458245853      0.17931144990
  16.198367905      0.46376317823
  5.0052600809      0.44171422662
   1  s  expopt
 0.58731856571       1.0000000000
   1  s  expopt
 0.18764592253       1.0000000000
   3  p    expopt  contopt
  13.571470233      0.40072398852E-01
  2.9257372874      0.21807045028
 0.79927750754      0.51294466049
   1  p  expopt
 0.21954348034       1.0000000000
#  1  d
# 1.0000000000       1.0000000000
*
```

File mos

```
$scfmo    scfconv=10   format(4d20.14)
# SCF energy is      -54.3329250250 a.u.  (virial theorem =  2.000000001)
#
     1  a1g    eigenvalue=-.15623888057347D+02   nsaos=3
-.99166890864040D+00-.28420294406651D-010.91519592317893D-02
     2  a1g    eigenvalue=-.92524548524703D+00   nsaos=3
0.30506869715453D+00-.65051761026701D+00-.44610487551870D+00
     3  a1g    eigenvalue=0.74881229854801D+00   nsaos=3
0.30759302935434D+00-.16295969601691D+010.16126161147521D+01
     1  t1u    eigenvalue=-.56865046629517D+00   nsaos=2
0.67926397018841D+000.46005039868410D+00
     2  t1u    eigenvalue=0.96169069264790D+00   nsaos=2
-.95675659621171D+000.10794148212163D+01
$end
```

15.6 ROHF of Two Open Shells

Extracts from `control` for O_2 in D_{3d} Symmetry

```
# HF-SCF/SVP


# Reference: triplet-sigma in D3d
# This is a Roothaan case (as is D-infinity-h).
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d3d
$closed shells
 a1g     1-3                        (2)
 a2u     1-2                        (2)
 eu      1                          (2)
$open shells type=1
 eg      1                          (1)
$roothaan         1
      a = 1      b = 2
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4774402753    149.4799190239   -298.9573592992


# Reference: singlet-delta in D3d
# This is a Roothaan case (as is D-infinity-h).
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d3d
$closed shells
 a1g     1-3                        (2)
 a2u     1-2                        (2)
 eu      1                          (2)
$open shells type=1
 eg      1                          (1)
$roothaan       1
      a = 1/2   b = 0
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4297623470    149.4298692899   -298.8596316369
```

Extracts from `control` for O_2 in D_{2h} Symmetry

```
# HF-SCF/SVP


# Triplet-sigma in D2h
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d2h
$closed shells
 ag      1-3                        ( 2 )
 b1u     1-2                        ( 2 )
 b2u     1                          ( 2 )
 b3u     1                          ( 2 )
$open shells type=1
 b2g     1                          ( 1 )
 b3g     1                          ( 1 )
$roothaan          1
      a = 1    b = 2
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4774402750    149.4798706643   -298.9573109393


# Singlet-delta in D2h : xx-yy component
# where x = b2g and y = b3g.  In D-infinity-h, b2g and b3g combine to eg.
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d2h
$closed shells
 ag      1-3                        ( 2 )
 b1u     1-2                        ( 2 )
 b2u     1                          ( 2 )
 b3u     1                          ( 2 )
$open shells type=1
 b2g     1                          ( 1 )
 b3g     1                          ( 1 )
$roothaan          2
$rohf
1b2g-1b3g    a = 0      b = 2
```

```
1b2g-1b2g    a = 1      b = 0
1b3g-1b3g    a = 1      b = 0
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4297623516    149.4298351805   -298.8595975321


# Singlet-delta in D2h : xy+yx component
# (an example of the general type: [xy]-singlet)
# where in D2h x = b2g and y = b3g are of different symmetry.
# In D-infinity-h, b2g and b3g combine to eg; see the reference
# calculation in D3d above.
#
$coord
   0.0             0.0              1.08597397921317       o
   0.0             0.0             -1.08597397921317       o
$symmetry d2h
$closed shells
 ag      1-3                        ( 2 )
 b1u     1-2                        ( 2 )
 b2u     1                          ( 2 )
 b3u     1                          ( 2 )
$open shells type=1
 b2g     1                          ( 1 )
 b3g     1                          ( 1 )
$roothaan          2
$rohf
1b2g-1b3g    a = 1      b = -2
1b2g-1b2g    a = 0      b =  0
1b3g-1b3g    a = 0      b =  0
$energy      SCF               SCFKIN            SCFPOT
     1   -149.4297623501    149.4298391833   -298.8596015334
```

Chapter 16

The Perl-based Test Suite Structure

16.1 General

Testing the TURBOMOLE modules for correctness and speed is the first task once the coding is completed. It is subject to automatization and thus requires a structure which is as simple and flexible as possible. In the Perl-based test suite this is implemented by a Perl script `TTEST` which performs all the testing and benchmarking tasks and resides in the central `scripts` directory of the TURBOMOLE installation. The test examples are located in subdirectories of the `TURBOTEST` directory, grouped according to the modules modules to be tested and a rough short/long classification. The benchmark suite shows the same directory structure and is rooted in the `TURBOBENCH` directory.

The central idea of the Perl-based test suite is that only the specific information about an individual test example is included in its local directory along with the input and reference files. This information is stored in the criteria file `CRIT` which contains the program calls, test criteria, and specific reference timings. Running the test script creates a new test subdirectory, usually called like `TESTDIR.i786-pc-linux-gnu`, where the TURBOMOLE programs are run and the results are summarized in the protocol file `TESTPROTOKOLL`.

16.2 Running the tests

Starting a single test example is simple. Change to the test example of your choice and call the `TTEST` script without arguments. The test is started in a subdirectory named `TESTDIR`.*sysname*, where *sysname* is the current platform name as returned by the `Sysname` script. The tested executable, a short description, and the test summary are output to the screen. Detailed information about the performed com-

mands and results of all test criteria are found in the TESTPROTOKOLL file in the test subdirectory.

The default location for the binaries and scripts used for testing is the $TURBODIR directory. If you like to test some other, e.g., your local version of the TURBOMOLE binaries or scripts, you can specify the loading paths by the -l or -ls options for the binaries and scripts, respectively,

```
TTEST -l /usr/local/TURBOMOLE/bin/i786-pc-linux-gnu \
   -ls /usr/local/TURBOMOLE/scripts.
```

A specific executable can be chosen by the -x option,

```
TTEST -x /usr/local/TURBOMOLE/bin/i786-pc-linux-gnu/dscf.
```

If a test output is already present, e.g., in the TESTDIR directory, you may wish to check the results. This is accomplished by calling TTEST in check mode,

```
TTEST --check TESTDIR,
```

which compares the results in TESTDIR with the reference and writes the results to the CHECKPROTOKOLL file in the test directory.

Testing parts of the TURBOTEST directory structure or the entire test suite at once is performed by calling the TTEST script from the appropriate place. The test script works recursively, executing all test examples underneath its starting directory. This requires that the test examples be arranged in a TURBOTEST-like directory structure,

```
progname/short|long/example  (e.g., dscf/short/H2O.SCF.E1),
```

and the TURBOTEST directory contain a DEFCRIT file with general test suite settings. If TTEST is started in the central TURBOTEST without any options, all available test examples are executed. By giving the list of module names (for full list, check TTEST --help) as argument to the script, the test can be restricted to these modules. The --short and --long options allow the user to select only the short or long test examples, respectively. Some examples of usage are given in the following table:

`TTEST dscf`	called in the TURBOTEST directory, performs only the tests for DSCF module.
`TTEST`	called in the TURBOTEST/dscf directory, does the same.
`TTEST --long`	executes long examples for all modules.
`TTEST ridft --short`	performs all short examples from the ridft directory.

Recursive testing creates some additional files in the central TURBOTEST directory. The global protocol file TESTPROTOKOLL.*sysname* contains short result messages for

all test and a list of errors occurred. The list of failed tests is also written to the PROBLEMS.*sysname* file and can be rerun by calling the test script with the `-r` option,

```
TTEST -r PROBLEMS.i786-pc-linux-gnu.
```

The `-r` may also be useful to create any user-defined selection of test examples. The full list of available examples is obtained by the `TTEST --list` command.

Once you are done with testing, you may wish to clean up afterwards. To do it, use the `--clean` and `--realclean` options of the `TTEST` script. The difference between these two is that `TTEST --clean` deletes only the test directories and protocols that were created for the current computer architecture as returned by `Sysname`. In contrast, the `TTEST --realclean` wipes out all test directories and protocols that get in its way.

16.3 Taking the timings and benchmarking

Benchmarking differs from testing only in that program timings are computed and compared with reference timings. Calling the script as

```
TTEST --timings
```

performs the test, calculates the CPU and wall clock timings, and writes the raw results to the TESTTIMINGS.*sysname.nodename* file. Auxiliary scripts `Tbtim` and `Tblist` help to convert this data to a more readable form and produce summaries as LaTeX tables. The `Tbtim` script creates a summary of benchmark results for a given computer platform from the original timings file. `Tblist` produces benchmark comparisons of different platforms. The corresponding timings files must be provided as arguments to the `Tblist` script. For more details and options, see `TBTIM --help` and `TBLIST --help`.

16.4 Modes and options of the TTEST script

The `TTEST` script knows several operation modes: "run", "check", "list", "clean", "realclean", and "validate", controlled by its options. The "run" mode is default and means that the test calculations are performed and the results are written to the `TESTPROTOKOLL` file. The "check" mode differs only in that the programs are not executed, but the existing program output is checked against the reference. The results of the check are written to the `CHECKPROTOKOLL` file. Calling the test script in the "list" mode simply lists the test examples that are currently available. This allows the user to save the full list to file, edit, and re-use it with the `-r` option. The "clean" and "realclean" options are for cleaning up the test directories and protocols. Finally, the "validate" mode is mainly of use for writing the `CRIT` files. It helps to verify the match patterns provided in the test criteria and shows if it extracts the

expected data for comparison with the reference. For every output file used for testing, the "validate" option produces a copy with an additional `.val` extension. The match strings evaluated for test criteria are highlighted in the output by `<<<<<` and `>>>>>` marks.

There is a lot of options controlling the behavior of `TTEST`Testing specific versions of TURBOMOLE modules is provided by loading path options, `-l` for binaries, `-ls` for scripts, and `-x` for a specific executable. For benchmarking, you need the `--timings` option to produce the timing summaries, and the `--newref` option to save the current program timings as the new reference. The module specifications and `--short`, `--long`, and `-r` options can be used for selecting the test examples. The more specialized options are summarized in the following table. Note that most of these options can also be set in the `DEFCRIT` file (see below).

Operation modes

`--help`
`-h`
`-?`
	Prints out the help message and exits.

`--list`	Lists the available test examples.
`--clean`	Deletes the test directories and summary files for the current architecture (given by SYSNAME, see Chapter 1.5).
`--realclean`	Deletes all test directories and protocols.
`--check` *dir*	Checks the correctness of an existing program test in the directory *dir* (default: `TESTDIR`.*sysname*). Useful if new criteria or new references are established.
`--validate` *dir* `-val` *dir*	Examines the output files in the directory *dir* (default: `TESTDIR`.*sysname*) and highlights the positions of the retrieved matches.

Loading path and naming options

`--loaddir` *dir* `-l` *dir*	Loading path for the TURBOMOLE binaries (default: `$TURBODIR/bin/`*sysname*).
`--scriptdir` *dir* `-ls` *dir*	Loading path for the TURBOMOLE scripts (default: `$TURBODIR/scripts`).
`--testprog` *prog* `-x` *prog*	Tests the given executable *prog*.
`--dir` *dir*	Name for the local test directory (default: `TESTDIR`.*sysname*).
`--critfile` *file*	Name for the local criteria file (default: `CRIT`).
`--defcritfile` *file*	Name for the test suite settings file (default: `DEFCRIT`).

`--protfile` *file* `--output` *file*	Name for the local protocol file (default: `TESTPROTOKOLL`).
`--gprotfile` *file*	Name for the global protocol file (default: `TESTPROTOKOLL`.*sysname*).
`--checkfile` *file*	Name for the check protocol file (default: `CHECKPROTOKOLL`).
`--errfile` *file*	Name for the local error output file (default: `output.err`).
`--probfile` *file*	Name for the failed tests list (default: `PROBLEMS`.*sysname*).
`--timfile` *file*	Name for the timings file (default: `TIMINGS`.*sysname*).
`--valfile` *file*	Name for the validation file for 'run' criteria (default: `RUNCRIT.val`).

Execution options

`--short` `--long`	Only `short` / `long` subdirectories of the test suite will be tested (default: `--short --long`).
`--restart` *file* `-r` *file*	The list of test examples for execution will be read in from *file* (default: `PROBLEMS`.*sysname*).
`--newref` *string*	Produces new reference timings and writes them to the `CRIT` file. A short description of the reference platform is provided by *string*.
`--fileref`	Produces new reference files.
`--batchmode`	Running in batch mode, no screen output.
`--errorstop` `--noerrorstop`	Stops / Does not stop after the first error. (default: `--noerrorstop`).
`--timings` `--notimings`	Writes / Does not write the timings on file for further processing. (default: `--notimings`).
`--runopts` `-o`	Sets the conditions under which the test is run (default: `"sequential, parallel"`)

Bibliography

[1] R. Ahlrichs; M. Bär; M. Häser; H. Horn; C. Kölmel. Electronic structure calculations on workstation computers: The program system Turbomole. *Chem. Phys. Lett.*, **162**(3), 165–169, (1989).

[2] A. Schäfer; H. Horn; R. Ahlrichs. Fully optimized contracted gaussian basis sets for atoms Li to Kr. *J. Chem. Phys.*, **97**(4), 2571–2577, (1992).

[3] A. Schäfer; C. Huber; R. Ahlrichs. Fully optimized contracted gaussian basis sets of triple zeta valence quality for atoms Li to Kr. *J. Chem. Phys.*, **100**(8), 5829–5835, (1994).

[4] K. Eichkorn; F. Weigend; O. Treutler; R. Ahlrichs. Auxiliary basis sets for main row atoms and transition metals and their use to approximate coulomb potentials. *Theor. Chem. Acc.*, **97**(1–4), 119–124, (1997).

[5] F. Weigend; F. Furche; R. Ahlrichs. Gaussian basis sets of quadruple zeta valence quality for atoms H–Kr. *J. Chem. Phys.*, **119**(24), 12753–12762, (2003).

[6] F. Weigend; R. Ahlrichs. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design an assessment of accuracy. *Phys. Chem. Chem. Phys.*, **7**(18), 3297–3305, (2005).

[7] A. K. Rappé; C. J. Casewit; K. S. Colwell; W. A. Goddard III; W. M. Skiff. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.*, **114**(25), 10024–10035, (1992).

[8] F. Weigend; M. Häser. RI-MP2: first derivatives and global consistency. *Theor. Chem. Acc.*, **97**(1–4), 331–340, (1997).

[9] F. Weigend; M. Häser; H. Patzelt; R. Ahlrichs. RI-MP2: Optimized auxiliary basis sets and demonstration of efficiency. *Chem. Phys. Letters*, **294**(1–3), 143–152, (1998).

[10] C. Hättig; F. Weigend. CC2 excitation energy calculations on large molecules using the resolution of the identity approximation. *J. Chem. Phys.*, **113**(13), 5154–5161, (2000).

[11] C. Hättig; K. Hald. Implementation of RI-CC2 for triplet excitation energies with an application to *trans*-azobenzene. *Phys. Chem. Chem. Phys.*, **4**(11), 2111–2118, (2002).

[12] C. Hättig; A. Köhn; K. Hald. First–order properties for triplet excited states in the approximated coupled cluster model CC2 using an explicitly spin coupled basis. *J. Chem. Phys.*, **116**(13), 5401–5410, (2002).

[13] C. Hättig. Geometry optimizations with the coupled-cluster model CC2 using the resolution-of-the-identity approximation. *J. Chem. Phys.*, **118**(17), 7751–7761, (2003).

[14] R. Bauernschmitt; R. Ahlrichs. Treatment of electronic excitations within the adiabatic approximation of time dependent density functional theory. *Chem. Phys. Lett.*, **256**(4–5), 454–464, (1996).

[15] R. Bauernschmitt; R. Ahlrichs. Stability analysis for solutions of the closed shell Kohn-Sham equation. *J. Chem. Phys.*, **104**(22), 9047–9052, (1996).

[16] F. Furche; R. Ahlrichs. Adiabatic time-dependent density functional methods for excited state properties. *J. Chem. Phys.*, **117**(16), 7433–7447, (2002).

[17] M. Kollwitz; J. Gauss. A direct implementation of the GIAO-MBPT(2) method for calculating NMR chemical shifts. Application to the naphthalenium and and anthracenium ions. *Chem. Phys. Lett.*, **260**(5–6), 639–646, (1996).

[18] M. von Arnim; R. Ahlrichs. Geometry optimization in generalized natural internal coordinates. *J. Chem. Phys.*, **111**(20), 9183–9190, (1999).

[19] P. Pulay; G. Fogarasi; F. Pang; J. E. Boggs. Systematic ab initio gradient calculation of molecular geometries, force constants, and dipole moment derivatives. *J. Am. Chem. Soc.*, **101**(10), 2550–2560, (1979).

[20] M. Dolg; U. Wedig; H. Stoll; H. Preuß. Energy-adjusted ab initio pseudopotentials for the first row transition elements. *J. Chem. Phys.*, **86**(2), 866–872, (1986).

[21] C. C. J. Roothaan. Self-consistent field theory for open shells of electronic systems. *Rev. Mod. Phys.*, **32**(2), 179–185, (1960).

[22] R. Ahlrichs; F. Furche; S. Grimme. Comment on "Assessment of exchange correlation functionals". *Chem. Phys. Lett.*, **325**(1–3), 317–321, (2000).

[23] M. Sierka; A. Hogekamp; R. Ahlrichs. Fast evaluation of the coulomb potential for electron densities using multipole accelerated resolution of identity approximation. *J. Chem. Phys.*, **118**(20), 9136–9148, (2003).

[24] F. Weigend. A fully direct RI-HF algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency. *Phys. Chem. Chem. Phys.*, **4**(18), 4285–4291, (2002).

[25] R. Fletcher. *Practical Methods of Optimization. Unconstrained Optimization.* Band 1. Wiley: New York, 1980.

[26] T. Helgaker. Transition-state optimizations by trust-region image minimization. *Chem. Phys. Lett.*, **182**(5), 503–510, (1991).

[27] F. Jensen. Locating transition structures by mode following: A comparison of six methods on the Ar_8 Lennard-Jones potential. *J. Chem. Phys.*, **102**(17), 6706–6718, (1995).

[28] P. Császár; P. Pulay. Geometry optimization by direct inversion in the iterative subspace. *J. Mol. Struct.*, **114**, 31–34, (1984).

[29] R. Fletcher. A new approach to variable metric algorithms. *Comput. J.*, **13**(3), 317–322, (1970).

[30] H. B. Schlegel. Optimization of equilibrium geometries and transition structures. *J. Comput. Chem.*, **3**(2), 214–218, (1982).

[31] H. B. Schlegel. Estimating the hessian for gradient-type geometry optimizations. *Theor. Chim. Acta*, **66**(5), 333–340, (1984).

[32] M. Ehrig. Diplomarbeit. Master's thesis, Universität Karlsruhe, 1990.

[33] T. Koga; H. Kobayashi. Exponent optimization by uniform scaling technique. *J. Chem. Phys.*, **82**(3), 1437–1439, (1985).

[34] A. K. Rappé; W. A. Goddard III. Charge equilibration for molecular dynamics simulations. *J. Phys. Chem.*, **95**(8), 3358–3363, (1991).

[35] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. General considerations. *J. Inst. Math. Appl.*, **6**(1), 76–90, (1970).

[36] D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comput.*, **24**(109), 23–26, (1970).

[37] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Math. Comput.*, **24**(111), 647–656, (1970).

[38] P. Pulay. Convergence acceleration of iterative sequences. the case of SCF iteration. *Chem. Phys. Lett.*, **73**(2), 393–398, (1980).

[39] M. P. Allen; D. J. Tildesley. *Computer Simulation of Liquids.* Oxford University Press: Oxford, 1987.

[40] K. Eichkorn; O. Treutler; H. Öhm; M. Häser; R. Ahlrichs. Auxiliary basis sets to approximate coulomb potentials (erratum, 1995, **242**, 283). *Chem. Phys. Lett.*, **242**(6), 652–660, (1995).

[41] J. A. Pople; R. K. Nesbet. Self-consistent orbitals for radicals. *J. Chem. Phys.*, **22**(3), 571–572, (1954).

[42] J. Čižek; J. Paldus. Stability conditions for solutions of Hartree-Fock equations for atomic and molecular systems. application to pi-electron model of cyclic plyenes. *J. Chem. Phys.*, **47**(10), 3976–3985, (1967).

[43] P. A. M. Dirac. Quantum mechanics of many-electron systems. *Proc. Royal Soc. (London) A*, **123**(792), 714–733, (1929).

[44] J. C. Slater. A simplification of the Hartree-Fock method. *Phys. Rev.*, **81**(3), 385–390, (1951).

[45] S. Vosko; L. Wilk; M. Nusair. Accurate spin-dependent electron-liquid correlation energies for local spin density calculations: a critical analysis. *Can. J. Phys.*, **58**(8), 1200–1211, (1980).

[46] J. P. Perdew; Y. Wang. Accurate and simple analytic representation of the electron-gas correlation energy. *Phys. Rev. B*, **45**(23), 13244–13249, (1992).

[47] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behaviour. *Phys. Rev. A*, **38**(6), 3098–3100, (1988).

[48] C. Lee; W. Yang; R. G. Parr. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, **37**(2), 785–789, (1988).

[49] J. P. Perdew. Density-functional approximation for the correlation-energy of the inhomogenous electron gas. *Phys. Rev. B*, **33**(12), 8822–8824, (1986).

[50] J. P. Perdew; K. Burke; M. Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, **77**(18), 3865–3868, (1996).

[51] J. Tao; J. P. Perdew; V. N. Staroverov; G. E. Scuseria. Climbing the density functional ladder: Nonempirical meta–generalized gradient approximation designed for molecules and solids. *Phys. Rev. Lett.*, **91**(14), 146401, (2003).

[52] A. D. Becke. A new mixing of Hartree-Fock and local density-functional theories. *J. Chem. Phys.*, **98**(2), 1372–1377, (1993).

[53] A. D. Becke. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.*, **98**(7), 5648–5652, (1993).

[54] J. P. Perdew; M. Ernzerhof; K. Burke. Rationale for mixing exact exchange with density functional approximations. *J. Chem. Phys.*, **105**(22), 9982–9985, (1996).

[55] V. N. Staroverov; G. E. Scuseria; J. Tao; J. P. Perdew. Comparative assessment of a new nonempirical density functional: Molecules and hydrogen-bonded complexes. *J. Chem. Phys.*, **119**(23), 12129–12137, (2003).

[56] F. D. Sala; A. Görling. Efficient localized Hartree-Fock methods as effective exact-exchange Kohn-Sham methods for molecules. *J. Chem. Phys.*, **115**(13), 5718–5732, (2001).

[57] F. D. Sala; A. Görling. The asymptotic region of the Kohn-Sham exchange potential in molecules. *J. Chem. Phys.*, **116**(13), 5374–5388, (2002).

[58] S. Grimme. Semiempirical hybrid density functional with perturbative second-order correlation. *J. Chem. Phys.*, **124**, 034108, (2006).

[59] A. Görling; M. Levy. Correlation-energy functional and its high-density limit obtained from a coupling-constant perturbation expansion. *Phys. Rev. B*, **47**, 13105, (1993).

[60] A. Görling; M. Levy. Exact Kohn-Sham scheme based on perturbation theory. *Phys. Rev. A*, **50**, 196, (1994).

[61] M. K. Armbruster; F. Weigend; C. van Wüllen; W. Klopper. Self-consistent treatment of spin-orbit interactions with efficient hartree-fock and density functional methods. *Phys. Chem. Chem. Phys.*, **10**, 1748–1756, (2008).

[62] M. K. Armbruster; W. Klopper; F. Weigend. Basis-set extensions for two-component spin-orbit treatments of heavy elements. *Phys. Chem. Chem. Phys.*, **8**, 4862–4865, (2006).

[63] M. Reiher; A. Wolf. Exact decoupling of the dirac hamiltonian. i. general theory. *J. Chem. Phys.*, **121**, 2037–2047, (2004).

[64] M. Reiher; A. Wolf. Exact decoupling of the dirac hamiltonian. ii. the generalized douglas-kroll-hess transformation up to arbitrary order. *J. Chem. Phys.*, **121**, 10945–10956, (2004).

[65] M. Reiher. Douglas-kroll-hess theory: a relativistic electrons-only theory for chemistry. *Theor. Chem. Acc.*, **116**, 241–252, (2006).

[66] A. Wolf; M. Reiher; B. Hess. The generalized douglas-kroll transformation. *J. Chem. Phys.*, **117**, 9215–9226, (2002).

[67] M. Sierka; A. Burow; J. Döbler; J. Sauer. Point defects in CeO_2 and CaF_2 investigated using periodic electrostatic embedded cluster method. *Chem. Phys. Lett.*, Seite submitted, (2007).

[68] K. N. Kudin; G. E. Scuseria. A fast multipole method for periodic systems with arbitrary unit cell geometries. *Chem. Phys. Lett.*, **283**, 61–68, (1998).

[69] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, **64**, 253–287, (1921).

[70] J. Hepburn; G. Scoles; R. Penco. A simple but reliable method for the prediction of intermolecular potentials. *Chem. Phys. Lett.*, **36**, 451–456, (1975).

[71] R. Ahlrichs; R. Penco; G. Scoles. Intermolecular forces in simple systems. *Chem. Phys.*, **19**, 119–130, (1977).

[72] S. Grimme. Accurate description of van der waals complexes by density functional theory including empirical corrections. *J. Comput. Chem.*, **25**(12), 1463–1473, (2004).

[73] S. Grimme. Semiempirical ggc-type density functional constructed with a long-range dispersion contribution. *J. Comput. Chem.*, **27**(15), 1787–1799, (2006).

[74] T. Schwabe; S. Grimme. Double-hybrid density functionals with long-range dispersion corrections: higher accuracy and extended applicability. *Phys. Chem. Chem. Phys.*, **9**, 3397–3406, (2007).

[75] F. Furche; D. Rappoport. Density functional methods for excited states: equilibrium structure and electronic spectra. In M. Olivucci, Ed., *Computational Photochemistry*, Band 16 von *Computational and Theoretical Chemistry*, Kapitel III. Elsevier, Amsterdam, 2005.

[76] F. Furche. On the density matrix based approach to time-dependent density functional theory. *J. Chem. Phys.*, **114**(14), 5982–5992, (2001).

[77] F. Furche; K. Burke. Time-dependent density functional theory in quantum chemistry. *Annual Reports in Computational Chemistry*, **1**, 19–30, (2005).

[78] D. Rappoport; F. F. Excited states and photochemistry. In M. A. L. Marques; C. A. Ullrich; F. Nogueira; A. Rubio; K. Burke; E. K. U. Gross, Eds., *Time-Dependent Density Functional Theory*, Kapitel 22. Springer, 2005.

[79] S. Grimme; F. Furche; R. Ahlrichs. An improved method for density functional calculations of the frecuency-dependent optical rotation. *Chem. Phys. Lett.*, **361**(3–4), 321–328, (2002).

[80] H. Weiss; R. Ahlrichs; M. Häser. A direct algorithm for self-consistent-field linear response theory and application to C_{60}: Excitation energies, oscillator strengths, and frequency-dependent polarizabilities. *J. Chem. Phys.*, **99**(2), 1262–1270, (1993).

[81] D. Rappoport; F. Furche. Lagrangian approach to molecular vibrational raman intensities using time-dependent hybrid density functional theory. *J. Chem. Phys.*, **126**(20), 201104, (2007).

[82] F. Furche. *Dichtefunktionalmethoden für elektronisch angeregte Moleküle. Theorie–Implementierung–Anwendung.* PhD thesis, Universität Karlsruhe, 2002.

[83] E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comp. Phys.*, **17**(1), 87–94, (1975).

[84] F. Haase; R. Ahlrichs. Semidirect MP2 gradient evaluation on workstation computers: The MPGRAD program. *J. Comp. Chem.*, **14**(8), 907–912, (1993).

[85] F. Weigend; A. Köhn; C. Hättig. Efficient use of the correlation consistent basis sets in resolution of the identity MP2 calculations. *J. Chem. Phys.*, **116**(8), 3175–3183, (2001).

[86] O. Christiansen; H. Koch; P. Jørgensen. The second-order approximate coupled cluster singles and doubles model CC2. *Chem. Phys. Lett.*, **243**(5–6), 409–418, (1995).

[87] W. Klopper; F. R. Manby; S. Ten-no; E. F. Valeev. R12 methods in explicitly correlated molecular electronic structure theory. *Int. Rev. Phys. Chem.*, **25**(3), 427–468, (2006).

[88] W. Klopper; B. Ruscic; D. P. Tew; F. A. Bischoff; S. Wolfsegger. Atomization energies from coupled-cluster calculations augmented with explicitly-correlated perturbation theory. *Chem. Phys.*, **356**(1–3), 14–24, (2009).

[89] C. Hättig; A. Köhn. Transition moments and excited state first-order properties in the second-order coupled cluster model CC2 using the resolution of the identity approximation. *J. Chem. Phys.*, **117**(15), 6939–6951, (2002).

[90] C. L. Janssen; I. M. B. Nielsen. New diagnostics for coupled-cluster and Møller-Plesset perturbation theory. *Chem. Phys. Lett.*, **290**(4–6), 423–430, (1998).

[91] I. M. B. Nielsen; C. L. Janssen. Double-substitution-based diagnostics for coupled-cluster and Møller-Plesset perturbation theory. *Chem. Phys. Lett.*, **310**(5–6), 568–576, (1999).

[92] T. Helgaker; P. Jørgensen; J. Olsen. *Molecular Electronic-Structure Theory.* Wiley: New York, 2000.

[93] O. Christiansen; P. Jørgensen; C. Hättig. Response functions from Fourier component variational perturbation theory applied to a time-averaged quasienergy. *Int. J. Quantum Chem.*, **68**(1), 1–52, (1998).

[94] C. Hättig; P. Jørgensen. Derivation of coupled cluster excited states response functions and multiphoton transition moments between two excited states as derivatives of variational functionals. *J. Chem. Phys.*, **109**(21), 9219–9236, (1998).

[95] A. Köhn; C. Hättig. Analytic gradients for excited states in the coupled-cluster model CC2 employing the resolution-of-the-identity approximation. *J. Chem. Phys.*, **119**(10), 5021–5036, (2003).

[96] C. Hättig; O. Christiansen; P. Jørgensen. Multiphoton transition moments and absorption cross section in coupled cluster response theory employing variational transition moment functionals. *J. Chem. Phys.*, **108**(20), 8331–8354, (1998).

[97] F. R. Manby. Density fitting in second-order linear-r_{12} Møller-Plesset perturbation theory. *J. Chem. Phys.*, **119**(9), 4607–4613, (2003).

[98] E. F. Valeev. Improving on the resolution of the identity in linear r12 ab initio theories. *Chem. Phys. Lett.*, **395**(4-6), 190–195, (2004).

[99] K. E. Yousaf; K. A. Peterson. Optimized auxiliary basis sets for explicitly correlated methods. *J. Chem. Phys.*, **129**(18), 184108, (2008).

[100] K. A. Peterson; T. B. Adler; H.-J. Werner. Systematically convergent basis sets for explicitly correlated wavefunctions: The atoms h, he, b–ne, and al–ar. *J. Chem. Phys.*, **128**(8), 084102, (2008).

[101] W. Klopper; C. C. M. Samson. Explicitly correlated second-order Møller-Plesset methods with auxiliary basis sets. *J. Chem. Phys.*, **116**(15), 6397–6410, (2002).

[102] W. Klopper; W. Kutzelnigg. Møller-Plesset calculations taking care of the correlation cusp. *Chem. Phys. Lett.*, **134**(1), 17–22, (1987).

[103] S. Ten-no. Explicitly correlated second order perturbation theory: Introduction of a rational generator and numerical quadratures. *J. Chem. Phys.*, **121**(1), 117–129, (2004).

[104] S. F. Boys. Localized orbitals and localized adjustment functions. In P.-O. Löwdin, Ed., *Quantum Theory of Atoms, Molecules and the Solid State*, Seite 253. Academic Press, New York, 1966.

[105] J. Pipek; P. G. Mezey. A fast intrinsic localization procedure applicable for ab initio and semiempirical linear combination of atomic orbital wave functions. *J. Chem. Phys.*, **90**(9), 4916–4926, (1989).

[106] D. P. Tew; W. Klopper. New correlation factors for explicitly correlated electronic wave functions. *J. Chem. Phys.*, **123**(7), 074101, (2005).

[107] F. A. Bischoff; S. Höfener; A. Glöß; W. Klopper. Explicitly correlated second-order perturbation theory calculations on molecules containing heavy main-group elements. *Theor. Chem. Acc.*, **121**(1), 11–19, (2008).

[108] S. Höfener; F. A. Bischoff; A. Glöß; W. Klopper. Slater-type geminals in explicitly-correlated perturbation theory: application to *n*-alkanols and analysis of errors and basis-set requirements. *Phys. Chem. Chem. Phys.*, **10**(23), 3390–3399, (2008).

[109] A. Hellweg; S. Grün; C. Hättig. Benchmarking the performance of spin–component scaled CC2 in ground and electronically excited states. *Phys. Chem. Chem. Phys.*, **10**, 4119–4127, (2008).

[110] C. Hättig. Structure optimizations for excited states with correlated second–order methods: CC2 and ADC(2). *Adv. Quant. Chem.*, **50**, 37–60, (2005).

[111] S. Grimme; E. Ugorodina. Calculation of 0-0 excitation energies of organic molecules by CIS(D) quantum chemical methods. *Chem. Phys.*, **305**, 223–230, (2004).

[112] Y. M. Rhee; M. Head-Gordon. Scaled second–order perturbation corrections to configuration interaction singles: Efficient and reliable excitation energy methods. *J. Phys. Chem. A*, **111**, 5314–5326, (2007).

[113] P. Deglmann; F. Furche; R. Ahlrichs. An efficient implementation of second analytical derivatives for density functional methods. *Chem. Phys. Lett.*, **362**(5–6), 511–518, (2002).

[114] P. Deglmann; F. Furche. Efficient characterization of stationary points on potential energy surfaces. *J. Chem. Phys.*, **117**(21), 9535–9538, (2002).

[115] M. Häser; R. Ahlrichs; H. P. Baron; P. Weis; H. Horn. Direct computation of second-order SCF properties of large molecules on workstation computers with an application to large carbon clusters. *Theor. Chim. Acta*, **83**(5–6), 455–470, (1992).

[116] T. Ziegler; G. Schreckenbach. Calculation of NMR shielding tensors using gauge-including atomic orbitals and modern density functional theory. *J. Phys. Chem.*, **99**(2), 606–611, (1995).

[117] A. E. Reed; R. B. Weinstock; F. Weinhold. Natural population analysis. *J. Chem. Phys.*, **83**(2), 735–746, (1985).

[118] C. Ehrhardt; R. Ahlrichs. Population analysis based on occupation numbers. ii. relationship between shared electron numbers and bond energies and characterization of hypervalent contributions. *Theor. Chim. Acta*, **68**(3), 231–245, (1985).

[119] F. Weigend; C. Schrodt. Atom-type assignment in molecule and clusters by pertubation theory— A complement to X-ray structure analysis. *Chem. Eur. J.*, **11**(12), 3559–3564, (2005).

[120] A. Klamt; G. Schüürmann. COSMO: A new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient. *J. Chem. Soc. Perkin Trans.2*, (5), 799–805, (1993).

[121] A. Klamt; V. Jonas. Treatment of the outlying charge in continuum solvation models. *J. Chem. Phys.*, **105**(22), 9972–9981, (1996).

[122] A. Klamt. Calculation of UV/Vis spectra in solution. *J. Phys. Chem.*, **100**(9), 3349–3353, (1996).

[123] F. J. Olivares del Valle; J. Tomasi. Electron correlation and solvation effects. I. Basic formulation and preliminary attempt to include the electron correlation in the quantum mechanical polarizable continuum model so as to study solvation phenomena. *Chem. Phys.*, **150**(2), 139–150, (1991).

[124] J. G. Ángyán. Rayleigh-Schrödinger perturbation theory for nonlinear Schrödinger equations with linear perturbation. *Int. J. Quantum Chem.*, **47**(6), 469–483, (1993).

[125] J. G. Ángyán. Choosing between alternative MP2 algorithms in the self-consistent reaction field theory of solvent effects. *Chem. Phys. Lett.*, **241**(1–2), 51–56, (1995).

[126] R. Cammi; B. Mennucci; J. Tomasi. Second-order Møller-Plesset analytical derivatives for the polarizable continuum model using the relaxed density approach. *J. Phys. Chem. A*, **103**(45), 9100–9108, (1999).

[127] O. Treutler; R. Ahlrichs. Efficient molecular numerical integration schemes. *J. Chem. Phys.*, **102**(1), 346–354, (1995).

[128] A. D. Becke. A multicenter numerical integration scheme for polyatomic molecules. *J. Chem. Phys.*, **88**(4), 2547–2553, (1988).

Index

(non)-append mode, 54
*, 37
-central, 185
-fanal, 169
-frznuclei, 184, 185
-relax, 87, 107
.map, 169, 277
.sys.data, 44
$2e-ints´_shell_statistics, 286, 287
$2e-ints_shell_statistics, 286
$D2-diagnostic, 256
$TURBODIR/uff/parms.in, 208
$actual step, 188
$alpha shells, 123, 142, 205, 226
$anadens, 168, 169
$atoms, 55, 128, 132, 203, 204, 228, 240, 246
$barrier, 281
$basis, 55, 94, 98, 202, 264
$beta shells, 123, 142, 205, 226
$boys, 267
$c1algorithm, 246
$cbas, 149, 154, 246, 247, 257
$cbasopt, 246
$cc2_natocc, 256
$cdspectrum, 144, 243, 247
$cgrad, 256, 257
$closed shells, 57, 58, 123, 204, 218, 225, 226
$constraints, 281
$coord, 43, 45, 94, 97–99, 126, 128, 131, 184, 191, 202, 207–209, 264
$coordinateupdate, 94, 258
    dqmax, 258
    interpolate, 258
    statistics, 258
$corrgrad, 263
$cosmo, 234–237
    allocate_nps, 234
    ampran, 234
    cavity, 234
      closed, 234
      open, 234
    disex, 234
    epsilon, 234
    nppa, 234
    nspa, 234
    phsran, 234
    routf, 234
    rsolv, 234
    use_old_amat, 234
$cosmo_atoms, 234–236
$cosmo_isodens, 236
$csconv, 284
$csconvatom, 284
$csmp2, 187, 283
$current, 279
$denconv, 29, 141, 149, 150, 154, 179, 211, 241, 247
$dft, 28, 122, 140, 182, 211, 226, 229, 232, 238, 241
    batchsize, 215
    functional, 211
    debug, 212
    dgrenze, 215
    diffuse, 213
    fgrenze, 215
    fullshell, 215
    functional, 226
    gridordering, 216
    gridsize, 212, 226
    gridtype, 212
    nkk, 212
    nphi, 212
    ntheta, 212
    old_RbCs_xi, 213

qgrenze, 215
radsize, 213
reference, 214
rhostart, 214
rhostop, 214
sgrenze, 215
symblock1, 215
symblock2, 215
test-integ, 215, 229
weight derivatives, 216
$dkhorder, 124, 232
$dkhorder 2, 124
$dkhparam, 124, 232
$dkhparam 1, 124
$dkhparam 2, 124
$dkhparam 3, 124
$dkhparam 4, 124
$dkhparam 5, 124
$drvopt, 72, 183, 237, 238
basis on, 98
$drvtol, 72
$ecp, 140, 202
$egrad, 94, 98, 257, 263, 265
$electrostatic field, 140, 216, 217
$embed, 126, 127, 129, 130, 232, 234
cell, 234
charges, 234
cluster, 234
content, 234
epsilon, 234
lmaxmom, 234
periodic, 234
potval, 234
wsicl, 234
$end, 43, 203
$energy, 123, 145, 203, 250, 254
$escfiterlimit, 243
$esp_fit, 275
$excitations, 154, 155, 161, 166, 171, 252, 255
bothsides, 252
conv, 252
exprop, 166, 252
irrep, 252
leftopt, 252
preopt, 252
spectrum, 171, 252
thrdiis, 252
tmexc, 171
xgrad, 252
$exopt, 144, 145, 244
$fermi, 71, 216
hlcrt, 216
nue, 216
stop, 216
tmend, 216
tmfac, 216
tmstrt, 216
$firstorder, 217
$fldopt, 140, 216, 217
1st derivative, 217
2nd derivative, 217
edelt, 217
fields, 217
geofield, 217
$forceapprox, 96–100, 203, 258, 261, 263, 264
format, 263
$forceconv, 239, 240
$forceinit, 41, 261, 264
diag, 262
carthess, 262
default, 262
individual, 262
off, 97, 261
on, 41, 97, 100, 258, 261, 264
carthess, 41, 101, 264
diag, 100
$forceiterlimit, 239, 240
$forcestatic, 264
$forceupdate, 97, 259, 264
ahlrichs, 259
indgeo, 259
maxgeo, 259
numgeo, 259
allow, 261
bfgs, 259
damping, 261
dfp, 259
dfp-bfgs, 259
diagonal, 261
ms, 259

offdamp, 261
offreset, 261
pulay, 260, 264
fail, 260
maxpul, 260
minpul, 260
modus, 260
numpul, 260
reseig, 261
scale, 261
schlegel, 259
thrbig, 261
threig, 261
$freeze, 140, 149, 150, 154, 179, 244, 246, 247
$gdiis, 123, 232
$gdiishistory, 258
$global, 94, 99, 262, 264
$globgrad, 94, 99, 263
$grad, 94, 97–99, 191, 203, 250, 254, 263, 265
$grad_send_dens, 287
$grid, 85, 270
$h0hessian, 91
$hessian, 72, 94, 100, 101, 238, 239, 262
$hessian (projected), 72, 264
$incore, 36, 217
$intdef, 43, 45, 96, 97, 183, 202, 257, 258, 262, 263
$interconversion, 92, 258
maxiter, 258
on, 96, 258, 259
qconv, 258
$ironly, 240
$isopts, 240
$isosub, 239, 240
$jbas, 111, 203, 227
$jkbas, 111, 228
$ke_control, 280, 282
$kramers, 123, 232
$laplace, 179, 247
conv, 247
$last MP2 energy change, 262
$last SCF energy change, 262
$last excitation energy change, 145
$last step
relax, 202
$lcg, 252
$les, 91, 182, 240
all, 240
$lesiterlimit, 240
$lhf, 230
$localize, 190, 275, 277
mo, 275
sweeps, 275
thrcont, 275
$lock off, 201
$loewdin, 268
$log, 279
$log_history, 280, 282
$m-matrix, 100, 262
$mao, 269
$mao selection, 274
$marij, 111, 228
extmax, 228
lmaxmom, 228
nbinmax, 228
precision, 228
thrmom, 228
wsindex, 228
$maxcor, 36, 70, 149, 154, 182, 239, 244–247
$maximum norm of
basis set gradient, 265
cartesian gradient, 265
internal gradient, 265
$md_action, 282, 283
$md_status, 279, 282
$mo output format, 218, 223
$mo-diagram, 218
$mointunit, 149, 150, 187, 245, 284
$moments, 267, 271
$moprint, 218
$mp2energy, 29, 149, 244, 246
$mp2energy SCS, 244
$mp2energy SCS pt=*val*1 ps=*val*2, 244
$mp2occ, 152, 246
$mp2pair, 246
$mulliken, 268
$mvd, 190, 271
$natoms, 279
$natural orbital

occupation, 203
$natural orbital occupation file=natural, 277
$natural orbitals, 203, 218
occupation, 218
$natural orbitals file=natural, 277
$newcoord, 181
$nmr, 187
dft, 187
mp2, 187
rhf, 187
shielding constants, 187
$nomw, 91, 239
$noproj, 239
$nosalc, 239
$nprhessian, 239
$nprvibrational normal modes, 239
$nprvibrational spectrum, 239
$nsteps, 279
$numprocs, 285
$oldgrad, 265
$open shells, 57, 58, 204, 225
$operating system, 201
$optimize, 92, 93, 183, 257–259
basis, 93, 257
logarithm, 257
scale, 257
cartesian, 93, 257
global, 93, 257
internal, 93, 96, 257, 262
redundant, 93, 257
$paboon, 268
$parallel_parameters, 286, 287
$parallel_platform, 34, 285
$pardft, 286
$path, 201
$point_charges, 140, 218
$points, 80, 81, 85, 267
$pointval, 169, 191, 192, 276
dens, 277
fld, 193, 277
fmt, 277
cub, 278
map, 278
plt, 278
txt, 278
vec, 278
xyz, 278
geo, 195, 278
line, 278
plane, 278
point, 278
integrate, 276
lmo, 194, 277
mo, 193, 277
nao, 195
nmo, 277
pot, 193, 277
$pop, 190, 272
atoms, 272
dos, 272
lall, 272
mo, 272
netto, 272
overlap, 272
thrpl, 272
$pop nbo, 190, 273
$pop paboon, 190, 273
$pople, 204
$prediag, 219, 221
$printlevel, 247, 250
$properties, 78, 267
$ramanonly, 240
$redund_inp, 205
$redundant, 97, 183, 257, 263
$response, 254
conv, 254
fop, 254
gradient, 254
nosemicano, 254
nozpreopt, 254
semicano, 254
thrsemi, 254
zconv, 254
zpreopt, 254
$response, 154, 165, 168, 179, 253, 255
$restart, 221
$restartd, 219, 221
$ricc2, 154, 155, 158, 159, 161, 165, 168, 175, 177, 179, 247, 248, 254, 255
adc(2), 248
cc2, 248

ccs, 248
cis, 248
cis(d), 248
cisdinf, 248
conv, 248
d1diag, 248
fmtprop, 248
geoopt, 165, 248
gsonly, 248
hard_restart, 248
iprint, 248
lindep, 248
maxiter, 248
maxred, 248
mp2, 248
mxdiis, 248
nohard_restart, 248
norestart, 248
oconv, 248
restart, 248
scs, 248
sos, 248
$ricore, 67, 110, 111, 140, 144, 182, 227, 228, 287
$ricore_slave, 287
$ridft, 140, 144, 227, 228
$rij, 122, 227, 232
$rik, 110, 122, 227, 228, 232
$ripop, 228
$rir12, 154, 171, 172, 175, 250, 251
ansatz, 251
cabs, 251
cabsingles, 251
comaprox, 251
corrfac, 251
examp, 251
local, 251
pairenergy, 251
r12model, 251
ump2fixed, 251
$rohf, 225
$roothaan, 204, 225
$rpaconv, 139, 242, 243
$rpacor, 144, 242
$rundimensions, 219
$scfconv, 29, 69, 141, 220, 224, 241, 247
settings for
AOFORCE, 182
NUMFORCE, 182
$scfdenapproxl, 217, 220
$scfdiis, 219, 221
$scfdump, 218, 219, 221
$scfinstab, 110, 139, 243
ciss, 241
cist, 241
dynpol, 242
non-real, 242
polly, 242
rpas, 241
rpat, 241
singlet, 242
triplet, 242
ucis, 241
urpa, 241
$scfintunit, 32, 36, 140, 219, 221, 222, 224, 247, 284
file, 222
size, 222
unit, 222
$scfiterinfo, 221
$scfiterlimit, 222
$scfmo, 58, 203, 204, 218, 219, 221–223, 226, 284
expanded, 223
file, 222
format, 223
none, 57, 222
scfconv, 223
scfdump, 223
$scfmo none, 57
$scforbitalorder, 223
$scforbitalshift, 223
automatic, 224
closedshell, 224
individual, 224
noautomatic, 224
$scftol, 224, 247, 284
$scratch
$scratch
files, 284, 286
$scratch files, 224, 262, 285
$seed, 280

$sharedtmpdir, 176
$soes, 140, 143, 144, 242–244
$soghf, 122, 123, 191, 232
$spectrum, 144, 243, 247
$spinor, 123
$start vector
    generation, 142, 243
$statistics, 222, 224, 225
    dscf, 111, 202, 224
    dscf parallel, 224, 286
    grad parallel, 287
    kora, 224
    mpgrad, 150, 202, 224, 245
    mpshift, 187
    off, 202, 224
    polly, 224
$statpt, 89, 91, 266
    bfgs, 91
    hssfreq, 266
    hssidiag, 266
    itrvec, 90, 266
    keeptmode, 266
    powell, 91
    radmax, 266
    radmin, 266
    threchange, 90
    thrmax-displ, 90
    thrmaxgrad, 90
    thrrmsdispl, 90
    thrrmsgrad, 90
    tradius, 89, 266
    update, 266
$sum
    rules, 243
$suspend off, 201
$symmetry, 140, 203
$thime, 109, 111, 140, 225, 247, 284
$thize, 109, 111, 140, 187, 218, 225, 247, 284
$title, 203, 280
$tmpdir, 176, 246, 247
$tplot, 152, 246
$traloop, 149, 150, 187, 245, 284, 285
$trand, 284
$trast, 284
$turbomole, 279
$twoint, 186
$uff, 102, 207
    maxcycle, 102
$uffgradient, 207, 209
$uffhessian, 207, 209
$ufftopology, 207–209
$uhf, 205, 226
$uhfmo_alpha, 142, 203, 222, 226
$uhfmo_beta, 142, 203, 222, 226
$userdefined bonds, 202
$vdw_fit, 270
$velocity gauge, 243
&, 37
plt, 276
NumForce
    -frznuclei, 184
-frznuclei
    NumForce, 184

Actual, 21
actual step
    dscf, 202
ADC(2)
    RI-, 247
analysis of normal modes
    internal coordinate, 183
Aoforce
    keywords, 238
aoforce, 12, 20, 21, 30, 70, 72, 77, 91, 94, 100, 137, 181–184, 238, 263, 264
Aoforce2g98, 21

B-matrix, 45, 46
babel, 26
Bend, 21
Boys localization, 275
Broken symmetry, 61
bsse_out, 108
bsseenergy, 106

Cbasopt, 21
cc1sd..., 169
cc1td-, 168
cc1td-cc2-gs-1a1-001, 169
cc1td-cc2-xs-3a2-001, 169
CC2, 19

RI-, 247
`CCL0--m-ss-xxx`, 165
`CCLE0-s--m-xxx`, 161
`CCME0-s--m-xxx`, 170
`CCNE0-s--m-xxx`, 166
`CCRE0-s--m-xxx`, 161
CCS
RI-, 247
`Cgnce`, 21
CIS, 19
RI-, 247
CIS(D), 19
RI-, 247
conjugate gradients, 92
`control`, 19, 21, 26, 28, 37, 39, 54, 57, 64, 90, 152, 168, 171, 176, 189
`converged`, 88
`coord`, 26
coordinates
frozen, 184
`cos`, 177, 250
COSMO
keywords, 234
`cosmo`, 21, 196–198, 234–237
`Cosmoprep`, 21, 236
counterpoise calculation, 54
CP-corrections, 106
CPHF, 186
`css`, 177, 250

`Define`, 151
`define`, 19, 23, 26, 28–30, 37–39, 41–46, 49, 54–58, 60–62, 64, 67, 73, 76, 78, 81, 82, 84, 85, 89, 90, 96, 97, 99, 106, 107, 110, 111, 123, 140, 143, 144, 149, 150, 154, 170, 172, 174, 183, 189, 194, 201, 202, 204, 222, 226, 228, 230, 241, 247, 258, 289
degrees of freedom, 45
`dens`, 169
DIIS, 92, 258
`Dist`, 22
`dos_a+b`, 272
`dos_a-b`, 272
`dos_alpha`, 272
`dos_beta`, 272
DSCF
keywords, 211
`Dscf`, 227
`dscf`, 12, 19, 20, 25, 28–36, 56, 57, 66, 88, 94, 105, 107, 109–113, 124, 125, 141, 150, 154, 155, 168, 176, 186, 189, 190, 192, 202, 203, 219, 224, 229, 230, 232, 234, 237, 270, 271, 284–287
dummy center, 43

`edens`, 169
EGRAD
keywords, 243
`egrad`, 12, 20, 21, 29, 30, 88, 94, 136, 137, 139–141, 144, 145, 168, 184, 189, 190, 192, 243, 263, 270, 271
`Eiger`, 22, 193, 275
`energy`, 101
environment
variable
OMP_NUM_THREADS, 35
ESCF
keywords, 241
`escf`, 12, 20, 29, 30, 69, 125, 136, 139–145, 232, 241–243
extended Hückel calculation, 56

`Freeh`, 30, 182
`freeh`, 21
FROG
keywords, 278
`frog`, 20, 30, 88, 105, 278, 279, 282, 283
frozen coordinates, 184

geometries
excited states, 165
ground state, 163
geometry
manipulation of, 50
GRAD
keywords, 237
`grad`, 19, 20, 28, 30–34, 66, 72, 88, 94, 97, 98, 105, 107, 109–113, 125, 145, 165, 176, 232, 234, 237, 243, 263, 265, 286, 287

grad_out, 108
gradient, 168
gradients
  excited states, 165
  ground state, 163

Hcore, 22

Infrared Spectra, 181
intense, 184
internal coordinates
  linear combination of, 49
  manual definition of, 48
  types of, 48
intersections
  conical, 160

jmol, 22
job.<cycle>, 88
job.last, 88
job.start, 88
jobbsse, 44, 106–108
Jobex, 22
JOBEX, 111
jobex, 21, 24, 27, 39, 67, 70, 87, 88, 90, 91, 101, 106, 137, 145, 154, 155, 250, 264, 287
  -c, 87, 107
  -dscf, 87
  -energy, 87, 107
  -ex, 87
  -gcart, 87, 107
  -grad, 87
  -gradient, 107
  -keep, 87
  -l, 87, 107
  -level, 87, 107
  -ls, 87, 107
  -md, 87
  -mdfile, 87
  -mdmaster, 87
  -mem, 107
  -opt, 107
  -relax, 87, 107
  -ri, 87, 107
  -rijk, 87
  -setup, 107
  -statpt, 87
  -trans, 87
  -trimer, 107

Kdg, 22
kinetic energy, 282

lalp, 275
lbet, 275
Leapfrog Verlet algorithm, 105, 279
Lhfprep, 22, 229
lhfprep, 229
lmo, 275
Log2egy, 22
Log2x, 22
LT-SOS-RI-MP2, 36, 178

mdens, 169
mdlog, 105
mdmaster, 278
mdmaster, 105
Mdprep, 22, 105, 278–280
mdprep, 278
menu
  atomic attributes, 50, 53
  general, 64, 65
  geometry main, 40
  geometry menu, 42
  internal coordinate, 45, 46
  occupation number assignment, 58, 59
  start vectors, 55, 56
molecular dynamics, 104, 278
molecular orbitals
  binary format, 56
MOLOCH
  keywords, 267
moloch, 78, 81, 82, 84, 85, 189, 267
MP2
  RI-, 247
Mp2prep, 22, 27, 29, 150
MPGRAD
  keywords, 244
mpgrad, 12, 19, 20, 29–32, 70, 88, 94, 98, 107, 147–151, 168, 188–190, 192, 202, 211, 224, 234, 237, 245, 263, 271, 285, 287

MPSHIFT
keywords, 283
`mpshift`, 12, 21, 27, 29, 30, 186–188, 285

`no weight derivatives`, 241
`not.converged`, 88, 108
`NumForce`, 21, 30, 31, 155, 184, 237
`Numforce`, 20–22, 137, 145, 155, 181, 182, 237, 238

OMP_NUM_THREADS, 35
OpenMP, 35
`Outp`, 22

parallelization
OpenMP, 35
threads, 35
`parms.in`, 208
PEECM
keywords, 232
plotting data
keywords, 271
`population analysis`, 272
properties
excited states, 165
ground state, 163

`q`, 37
quasi–Newton, 92

`Raman`, 22, 30, 184
Raman spectra, 184
RDGRAD
keywords, 237
`Rdgrad`, 227
`rdgrad`, 19, 20, 28, 30–32, 34, 67, 88, 94, 97, 107, 109–111, 113, 125, 145, 165, 232, 234, 237, 286, 287
RELAX
keywords, 257
`relax`, 12, 20, 21, 30, 41, 46, 73–75, 87, 88, 92–94, 96, 97, 99–101, 106–108, 257, 258, 261, 262, 264, 265
`restart.cc`, 249
RI-ADC(2), 247
RI-CC2, 19, 247
keywords, 247
RI-CCS, 247
RI-CIS, 19, 247
RI-CIS(D), 19, 247
RI-MP2, 247
RI-MP2-F12, 36
RICC2
keywords, 247
`ricc2`, 11, 12, 19, 21, 29–36, 52, 53, 70, 147–151, 153–156, 158–162, 165, 167–170, 172, 176–178, 189, 192, 247, 255, 263
RIDFT
keywords, 211
`Ridft`, 227
`ridft`, 12, 19, 20, 27–32, 34, 57, 67, 68, 88, 107, 109–111, 113, 122–125, 141, 154, 155, 168, 170, 186, 189, 190, 192, 194, 203, 227, 228, 232, 234, 237, 270–272, 286, 287
RIMP2
keywords, 244
`rimp2`, 12, 19, 20, 29, 30, 52, 53, 70, 88, 94, 107, 147–153, 158, 189, 190, 192, 245, 246, 248, 263, 270, 271
`Rimp2prep`, 27, 29, 149, 246, 247
Roothaan parameters, 61

`Screwer`, 22
`screwer`, 92
`scs`, 175, 177, 250
SCS-MP2, 177
`Sdg`, 22
Simulated Annealing, 283
`sos`, 177
SOS-MP2, 177
SOS-RI-MP2, 36
fourth-order scaling, 178
spectra
Raman, 184
spin
flipping spins on atoms, 58
`Stati`, 22, 111
STATPT
keywords, 265
`statpt`, 20, 30, 68, 87, 89–91, 106–108
steepest descent, 92

STOP, 88
stop, 88
structure library, 43
structure optimization, 87
substitution, 43
Sysname, 22, 24, 311, 313
sysname, 24, 25

t2x, 22, 191
Tblist, 23, 313
Tbtim, 23, 313
temperature, 282
time, 105, 279
timestep, 279
tm2molden, 22, 191
Tors, 23
transformation
    Laplace, 178
TTEST, 311–314
TURBOMOLE, 9, 11–13, 16, 21–26, 28–33, 37, 39, 42, 53, 54, 56, 64, 65, 87, 90, 92, 97, 106, 115, 125, 128, 132, 147, 151, 157, 181, 184, 191, 201, 219, 222, 223, 227, 232, 235, 236, 269, 279, 285, 289
TURBOMOLE
    installation, 24
    modules, 19
    quotation of, 11
    tools, 21
Turbotest, 25
twoint, 27, 32, 33

UFF
    keywords, 207
uff, 19, 41, 90, 91, 101, 102, 207, 208
uffgradient, 102
uffhessian0-0, 102
ufftopology, 102, 208, 209, 211
    nxtn12, 208
Uhfuse, 23

vector
    function, 157
velocity, 282
vibration, 181
Vibrational Frequencies, 181

wave function analysis
    keywords, 271

x2t, 23, 26, 39
xxx.map, 168